SALES_2: A multi-material extension to the SALE hydrocode with improved equation of state and constitutive model.

Gareth S. Collins and H. Jay Melosh

September 13, 2002

Foreword

Thank you for downloading SALES_2. This document is a not a complete user manual; it merely describes modifications made to the original SALE hydrocode and has been designed to complement the original SALE manual. A copy of the SALE manual is available at:

http://www.lpl.arizona.edu/tekton/sales_2.html.

SALES_2 is distributed as a number of related files:

sales_2.f The source code (Just over 4000 lines including comments!)

sales_2.comm The common block (included by most subroutines)

sales_2.input The input file (set up for an example calculation)

sales_2.pdf This manual

example.pdf Graphics output file from an example calculation

example.out Output file from an example calculation

Old users: please note that this document is distributed with version 2.1 of SALES_2.

SALES_2 also requires the PGPLOT graphics libraries. These can be downloaded from: http://www.astro.caltech.edu/~tjp/pgplot/.

The source code is heavily annotated, as it is expected that individual users will want to modify portions of it for their own particular problems. H. J. Melosh and his colleagues will attempt to answer technical questions, but cannot undertake the task of educating users in how to do hydrocode simulations.

Enjoy working with SALES_2. Any comments or suggestions are welcomed. Please contact the author: gareth@lpl.arizona.edu.

Contents

1 Introduction **4**

2 Overview **6**
2.1 Problem set-up 6
2.2 Cycle initialization 6
2.3 Lagrangian computation 8
2.4 Cycle completion 8

3 Improvements to the I/O routines **9**
3.1 Problem set-up improvements 9
3.1.1 New input file 9
3.1.2 Start information 9
3.2 Improvements to output routines 10
3.2.1 Plotting routines 10
3.2.2 Monitoring tracers 10
3.2.3 Output data files and dump files 12

4 Modifications to the SALE solution algorithm **14**
4.1 Problem Set-up 14
4.1.1 Mesh generation and multi-material capability 14
4.1.2 Resolution 15
4.1.3 Alternate Node Coupler 16
4.2 Equation of state 18
4.2.1 The liquid equation of state 18
4.2.2 The perfect gas equation of state 19
4.2.3 The Mie-Grunsien and Murnaghan equation of state . 19
4.2.4 The Tillotson equation of state 19
4.2.5 SALES_2 equation of state algorithm 21
4.3 Constitutive model 21
4.3.1 Viscosity and Newtonian fluid flow 23
4.3.2 Elasticity and the Kelvin Substance 23
4.3.3 Plasticity, Yield Strength and the Bingham Substance 24
4.3.4 Grady-Kipp fragmentation algorithm 26

5 Description of parameters **29**
5.1 Input quantities 29
5.2 Derived and scalar quantities in the common block 32
5.3 Cell and vertex variables 34

6 Compilation, running and a case example **36**
6.1 Compilation issues and portability 36
6.2 Example calculation 36

Chapter 1

Introduction

Hydrodynamic computer codes (hydrocodes) are powerful numerical tools for studying the dynamics of continuous media. Numerous hydrocodes exist and have been applied to a vast range of problems, from simple fluid flow to complex impact cratering simulations involving phase changes and multiple materials. However, there is no such thing as a universal code; each hydrocode is developed to study only a small subset of the infinite number of possible problems. Thus, careful attention must be paid when choosing the appropriate hydrocode for a given problem.

SALES_2 is an extensively modified version of the Simplified Arbitrary Lagrangian-Eulerian computer program SALE (Amsden et al., 1980). The original SALE code was developed to study only viscous fluid flow; for example, fluid flow through a channel or pipe. SALES_2 has retained all the functionality of the original SALE code; however, it can also model elastic and plastic deformation and tensile failure. Problems that have been addressed by SALES_2 (or a close facsimile of) include: crater collapse; dynamic fragmentation during an impact; spallation; and atmospheric breakup of meteoroids during atmospheric entry.

SALE is extremely tractable - comprising just over 4000 lines of code. This fact, together with its reliability and stability, have resulted in its continued use within the academic community for a wide range of problems. Despite its inherent suitability as an introduction to hydrocodes, however, newcomers to the field are recommended to investigate the subject area before attempting to use the code. For an excellent overview of the theory of hydrocodes please refer to Anderson (1987).

The modifications and extensions to SALE that are documented in this report make SALES_2 a versatile computer code. SALES_2 may be applied to problems involving materials of various types and rheologies, and study deformations over a range of speeds and external conditions. However, like any hydocode, SALES_2 cannot be applied to any problem. In particular, the Eulerian formulation within SALES_2 is restricted to one material only, which limits the use of SALES_2 for problems involving extreme deformation, such as the excavation stage of an impact event. It is strongly recommended, therefore, that budding hydrocode users ensure that SALES_2

is the appropriate tool for their problem, before embarking on installing or modifying the software.

This documentation provides only a brief overview of the SALES_2 solution algorithm, which is presented in chapter 2. The aim of the manual is to describe in detail the modifications made by Jay Melosh and his co-workers to the original version of SALE. Users of SALES_2 are thus referred to the original SALE manual to complement this report. The SALE manual is available online at:

http://www.lpl.arizona.edu/tekton/sales_2.html.

The primary modifications to SALE included in SALES_2 are two-fold. Firstly, improvements to the input and output routines make SALES_2 more user-friendly than its predecessor; and it has been modified to use the fortran plotting routines of PGPLOT. These modifications are discussed in chapter 3. Secondly, and more importantly, SALES_2 includes a far more sophisticated equation of state algorithm and a much more versatile constitutive model, which includes the Grady-Kipp fragmentation algorithm. Chapter 4 describes these modifications and, where appropriate, refers the user to the relevant literature for a more detailed explanation.

SALE was written in FORTRAN77, a complete listing of which accompanies the SALE manual. SALES_2 contains many modified subroutines and some additions also written in FORTRAN77. The complete FORTRAN77 listing for SALES_2 can be downloaded from the SALES_2 website:

http://www.lpl.arizona.edu/tekton/sales_2.html.

Included in the download should be a copy of this document, the code (sales_2.f), the common-block include file (sales_2.comm) and a sample input file (sales_2.input) with all the input parameters for a specific test problem.

We attempt to keep our code as well annotated and clean as possible, however, our use of the code has been almost exclusively on a Sun-Unix platform so we offer no guarantee on the portability of SALES_2. Compilation issues and an example calculation are discussed in the final chapter of this report. Any comments or technical questions are welcome: please contact the author (Email: gareth@lpl.arizona.edu).

Chapter 2

Overview

Figure 2.1 presents the SALES_2 solution algorithm for comparison with figure 4 on page 20 of the original SALE manual. Highlighted are the subroutines which have been modified in SALES_2. The algorithm is separated into four sections: Problem set-up, cycle initialization, Lagrangian computation and cycle completion. These sections are discussed below. Names of subroutines are in italic.

2.1 Problem set-up

In this part of the code the problem is initialized, be it an entirely new calculation or the resumption or restarting of an old calculation. First the code processes the input file. If necessary, it then restores previously stored information from a dump file, before computing some derived variables used throughout the calculation. If the problem is a new one the mesh is then generated. It is this part of the code (*celset*) where users will need to implement their own mesh generating algorithm for a specific problem. Mesh generation and the multiple material facility are discussed in section 4.1.1.

Finally, all the vertex and cell-centered quantities are initialized in *celset*. The initial pressures are computed using the equation of state via the *eos* routine, except for the incompressible case for which the initial pressure is zero. The subroutine *bcset* adjusts the boundary values if inflow or pressure boundaries are specified.

2.2 Cycle initialization

In this section, the code advances to the next time step. First the size of the time step is chosen according to the same rules as the original SALE. The cell pressures are then calculated for the compressible case using *eos* (discussed in section 4.2). Finally, outputs are performed when appropriate and the time and cycle number are incremented. At the end of the cycle the code returns to the beginning of this section of the code.

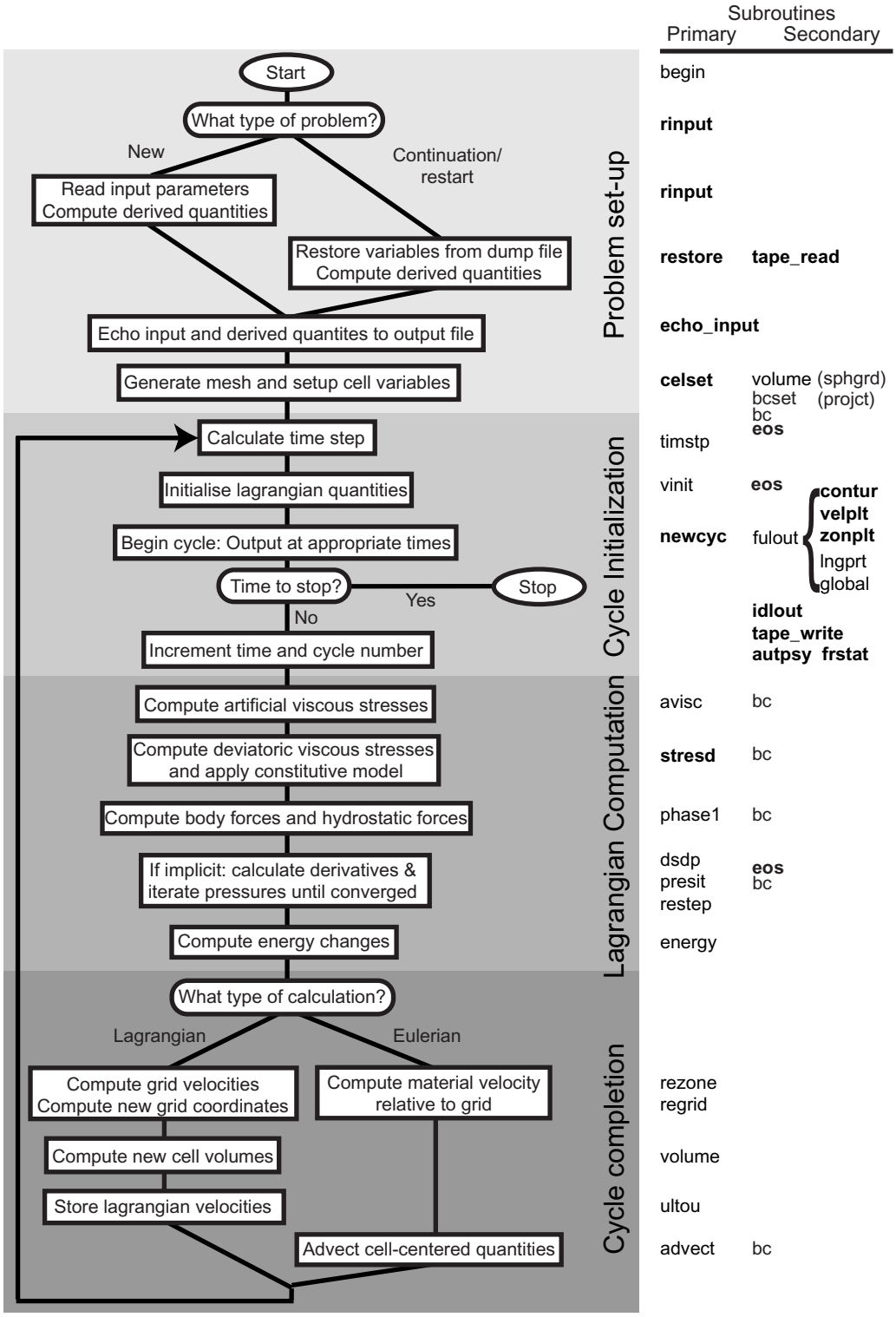


Figure 2.1: A general flow chart for the SALES_2 program

2.3 Lagrangian computation

It is this section of the code where the effect of all forces on the velocity field is determined. In addition, the changes in internal energy are computed. First the code computes the effect of any artificial forces applied in the subroutine *avisc*. These artificial forces allow the code to treat the passage of shock-waves through the mesh and guard against degenerate mesh instabilities. See the original SALE manual for further details. Secondly, the code computes the effect of viscous and elastic stresses in the *stresd* routine. It is here that the constitutive model capabilities of SALE have been extended. This is discussed in section 4.3. Finally, the effect of any external forces and the pressure forces are computed in subroutine *phase1*.

If the calculation is in the very low-speed regime, or incompressibilty is assumed, then the pressure forces may be determined by a Newton-Raphson iteration. This method offers a numerically stable means by which pressure signals can traverse more than one cell in a time step. Thus, time advanced pressures and velocities can be calculated with greater efficiency than using the explicit pressure calculations with a small time step.

The last calculation in this section of the code is the energy calculation which is done in subroutine *energy*. Here the contributions of the pressure work and viscous dissipation are combined to compute the change in the internal energy of the cell.

2.4 Cycle completion

Depending on the type of calculation (i.e. Lagrangian, Eulerian or other) this section of the code modifies the shape of the mesh or fluxes material through the mesh. In the purely Lagrangian case the new grid coordinates are computed (in subroutine *regrid*) using the combined velocities computed in the previous section. The cell volumes are then modified (subroutine *volume*) and the vertex velocities are stored for use next cycle (subroutine *ultou*). In the Eulerian case the velocity of the material in a cell relative to the fixed mesh is computed in subroutine *regrid*. The cell-centered quantities of internal energy, density and momentum are then moved through the mesh to adjacent cells in subroutine *advect* before the momenta are converted back to velocities. Another type of rezoning may be implemented in this section of the code if required. This should be done in the subroutine *rezone*. After this section of the code is completed the code returns to the beginning of the "Cycle Initialization" stage to begin another time step.

Chapter 3

Improvements to the I/O routines

This sections discusses the improvements made to the first two sections of the SALE hydrocode. It describes the improved input file and restart procedure, plus the modifications made to the output routines (both graphical and data). When quoted, the names of subroutines appear in italic and variable names appear in bold face.

3.1 Problem set-up improvements

3.1.1 New input file

The input file for SALES_2 (sales_2.input) is heavily annotated: each parameter is briefly described and options are listed for various integer flags. For a description of all the variables used in SALES_2, including the input parameters see section 5.

The input file is read by the subroutine *rinput*. Comments in the input file begin with the string "/∗". When reading the input file these comments are skipped using the function *skip*.

After computing some variables derived from the input parameters the subroutine *echo_input* writes the input parameters and some derived parameters to the output file to ensure the input file has been read correctly.

NB: *It is very important during code development for the user to examine the output and be sure that the code is reading what the user has input. Many errors have been made in the past because this simple step has been overlooked!*

3.1.2 Start information

jname: This variable is an eighty-character string for the problem title. This string will appear at the top of any output file and in the title of any graphics plots. The string should be enclosed in single quotes.

ibegcont: This integer variable defines whether the problem is a new problem (**ibegcont** = 0), a continuation of an old problem (1) or a restart of a previously terminated problem (2). If the problem is a new problem the entire input file will be read. If, however, the problem is a continuation or a restart the input file will only be read until after the time information (labelled in the input file). The rest of the input variables, as well as derived variables and cell and vertex quantities, are read from the dump file by the subroutine *tape_read*. The only difference between a continuation and a restart is that for a restart the problem time and cycle number are reset to zero instead of resuming at the problem time that the dump file was made.

ndump: This integer variable defines the number of the dump file from which the input parameters and cell and vertex quantities will be restored.

3.2 Improvements to output routines

3.2.1 Plotting routines

SALES_2 uses the FORTRAN77 graphics libraries of PGPLOT to produce some basic graphical representation of various vertex and cell-centered quantities during run time. The vertex positions are plotted using *zonplt* (in lagrangian mode only), the vertex velocities are plotted using *velplt* and various cell-centered quantities are contoured using *contur*.

NB: *SALES_2 will not compile or run unless PGPLOT has been installed and linked correctly. For more information on PGPLOT see Chapter 6.*

Before prompting the user for the input and output file names the code will request an output type for the plotting routines. The types of output available will depend on the drivers chosen when installing PGPLOT. The standard output types for a Unix system are: /Xwin - Output to an X-window on the screen; /ps - Output to a postscript file; /cps - Colour postscript; /null - suppress graphical output.

The plotting routines are called from *fulout*. The plotting routines are called at regular intervals of **dtplot** in problem time and at the zeroth and last time step.

3.2.2 Monitoring tracers

The plotting routines included in SALE, which in SALES_2 utilize PGPLOT, display properties of the mesh at regular time intervals (**dtplot**) during the run. It is frequently the case, however, that the user's interest is not in one particular instant in time, but in the behaviour of a particular cell or particle during the entire calculation. Such particles (or cells) are often called (Lagrangian) tracer particles. In order to facilitate the study of tracer particles a routine called *idlout* has been added to SALE. The routine outputs information about certain cells or vertices in the mesh to separate files for each cell or vertex.

NB: *We use IDL (Interactive Data Language) to interpret the output files produced by* idlout. *Due to the wide use of IDL we have decided to leave this routine in the released version of SALES_2. However, we have not released the IDL routines as they are quite problem specific. It is envisioned that users of SALES_2 will either disable the* idlout *routine (by setting* $\boldsymbol{iidlp} = 0$*) or modify this routine to make their own output files for analysis using IDL or some other plotting package.*

iidlp: This integer flag defines whether or not the *idlout* routine is called. Set **iidlp** = 1 to enable the *idlout* routine and **iidlp** = 0 to disable the routine. (Note, however, that when disabling *idlout* all the other information must be consistent. That is, if **ncells** is defined as 5, for example, there must be 5 sets of i, j coordinates defining those cells.)

dtidlp: This double precision variable defines the problem-time interval between writing to the IDL output files.

ncells: This integer variable defines the number of cells for which data will be written to the output files. SALES_2 is currently limited to output data for 10 cells only. This may be increased by modifying the *idlout* routine as directed in the comments.

nplots: This integer value defines the number of variables to be recorded for each of the cells. A maximum of 12 quantities may be recorded:

1. Pressure (cell)
2. Velocity in x-direction (particle)
3. Velocity in y-direction (particle)
4. Maximum principal stress (cell)
5. Intermediate principal stress (cell)
6. Minimum principal stress (cell)
7. Density (cell)
8. Mass (cell)
9. Volume (cell)
10. x-coordinate (particle)
11. y-coordinate (particle)
12. Damage (cell)

i_cell, j_cell: These integer arrays record the i and j locations of the cells to be traced. The i and j correspond to the cell quantity and the bottom left vertex of the cell. There should be **ncells** sets of i and j pairs.

plot_type: This integer array defines the quantities to be recorded for each cell, taken from the list above. There must be **nplots** number of elements in **plot_type**.

In summary, *idlout* records a number of selected cell-centered or particle properties for each cell/particle of interest during run time. Every interval of **dtidlp** seconds (problem time) these properties are written to an output file corresponding to each tracer particle. The output files may then be interpreted using various IDL plotting routines. The user is required to enter the number of tracer cells/particles (**ncells**) and the number of properties to be recorded (**nplots**). For each cell/particle desired the user must enter an i, j location. The user must also specify the properties required, chosen from a list of possibilities.

NB: *The labelling convention is the same as the original SALE: The i and j corresponding to the cell quantity is that which corresponds to the bottom left vertex of the cell.*

In addition to generating the actual output data files the *idlout* routine also creates a summary file, containing information regarding how many and which cells were traced, what properties were traced, how many data points were created, and so on. This information is required by the plotting routines in order to generate the plots of each property for every cell.

3.2.3 Output data files and dump files

SALES_2 also provides the facility to output all important cell-centered and vertex quantities to an output file "longprint.dat". Longprints are activated by setting the **ilprt** flag to 1 in the input file. Longprints also create a file: "global.dat" which provides a printout of the total internal energy in the mesh, the total energy and the total momentum. The problem-time interval separating long-printouts is defined by the input parameter: **dtlprt**. The subroutines which generate these output files are *lngprt* and *global*.

The last output type is the tape-dump. These dumps are forced by setting the **idump** flag to 1 and occur at problem-time intervals of **dtdump**. The dump file created contains all the cell-centered and vertex quantities for the entire mesh, input parameters and derived parameters necessary for restarting the simulation following termination. This is a useful feature if the user wishes to continue with an unstable calculation using a smaller time-step; continue a previously terminated calculation; or use a previous calculation as starting conditions for a new calculation.

The dump files are *unformatted* meaning that a human user cannot interpret them (this is to minimize the file size). Consequently, care must be taken to ensure that the tape dump routines (*tape_read* and *tape_write*) are

coded such that the order in which each variable is written to the dump file exactly matches the order in which they are read.

Chapter 4

Modifications to the SALE solution algorithm

This section discusses the extensions made to SALE that allow SALES_2 to model a variety of different materials and rheologies. First the multi-material capability is discussed. Then the equation of state and the constitutive model that complete the set of equations solved by SALES_2 are described in depth. Finally the fragmentation algorithm is briefly introduced.

4.1 Problem Set-up

4.1.1 Mesh generation and multi-material capability

The mesh generating algorithm in SALES_2 generates a rectangular mesh with the origin at the bottom-left corner. Two optional subroutines have been included: *sphgrd* which converts a rectangular grid into a spherical grid; and *projct* which deforms the top of the mesh to simulate a projectile having just made contact with the mesh. For more complicated mesh configurations the user will have to modify the algorithm directly in the subroutine *celset*.

Also in *celset* is the algorithm for assigning a material number for each cell. The multi-material facility does not allow for more than one material to occupy a single cell and hence is limited to purely Lagrangian calculations only.

The multi-material facility included in SALES_2 is very basic as it is envisioned that each user will wish to customize the algorithm extensively. The algorithm included in this version divides the mesh into **numat** equal layers, where **numat** is the number of materials desired (maximum 4 layers). Any change in the geometry or sizes of these layers should be made by modifying the "multi-material" part of the subroutine *celset*.

The algorithm is straightforward: The mesh is divided into layers by three boundary lines, **bndry1**, **bndry2** and **bndry3**; depending on how many materials are required any number of these may be superposed and placed at the top of the mesh such that they have no effect on dividing the mesh. For example, if a two-material mesh is desired then **bndry1** is set at

half-way up the mesh, while **bndry2** and **bndry3** are both set at the top of the mesh. The program then sweeps over every cell defining a material number, **imat**, for every cell depending on whether the cell is below **bndry1** or above. If all three boundaries are set within the mesh then a maximum number of 4 materials is defined.

Modifying this algorithm to include more than four material layers, or unequal layer heights should be intuitive, however, one should remember that while there are $\mathbf{nx} * \mathbf{ny}$ cells in the mesh there are $(\mathbf{nx}+1) * (\mathbf{ny}+1)$ vertices.

The number of materials required is set by the parameter **numat**, in the input file. A maximum of four materials is permitted (in the current configuration). The input file also requires a list of material descriptions (a string of 32 characters) to label the materials to be used. The remaining material input parameters require a value for each material selected. These parameters are discussed in the relevant sections below or in the original SALE manual.

4.1.2 Resolution

The choice of resolution in space and time is an important issue when using hydrocodes. A simulation should be conducted with a high enough resolution to resolve all the important flow variations in space and time. However, these needs must be balanced by the specifications of available hardware and the time available to run the simulation. The amount of memory required for a given simulation is roughly proportional to the number of cells in the mesh and the duration of the calculation is dependent on the processor speed and, more importantly, the size of the time step.

The choice of cell size is both hardware and problem specific and is left to the user to decide. The time step size, however, is often restricted by certain criteria, some of which are discussed in the original SALE manual. A good rule of thumb, however, is to use a time step that is as large as possible given the appropriate restrictions. In many cases, the largest appropriate time step size is limited by the Courant condition on sound signal propagation. This time step dt_{cou} is the minimum time taken for sound to cross a cell:

$$dt_{cou} < \min\left(\frac{dx}{c}, \frac{dy}{c}\right), \tag{4.1}$$

where dx and dy are the cell dimensions and c is the speed of sound in the material modelled. The minimum implies that every cell in the mesh must be considered to ensure that dt_{cou} satisfies the most restrictive case. SALES_2 stipulates that the time step is never larger than one-fifth of the minimum sound propagation time: $dt < 0.2dt_{cou}$.

It is also worth pointing out that, when conducting numerical analysis of physical phenomena, it is good practice to test the effect of resolution, in space and time, on the calculation results. If an observed result is independent of the density of mesh divisions and the frequency of the sampling time,

then it is unlikely to be a numerical artifice. Such resolution studies have identified a weakness in the alternate-node coupling scheme implemented in SALE; a modification to correct for this flaw is discussed next.

4.1.3 Alternate Node Coupler

In a Lagrangian calculation of fluid flow using quadrilateral mesh cells, there are certain mesh deformations that do not result in net pressure changes; that is, there is change in cell shape but no change in cell volume. Such nonphysical motion of vertices is sometimes termed "vertex coasting". Figure 4.1 illustrates two such patterns, the bow-tie pattern and the herringbone pattern. Typically, vertex coasting is associated with the shortest resolvable wavelengths ($2dx$) in the mesh.

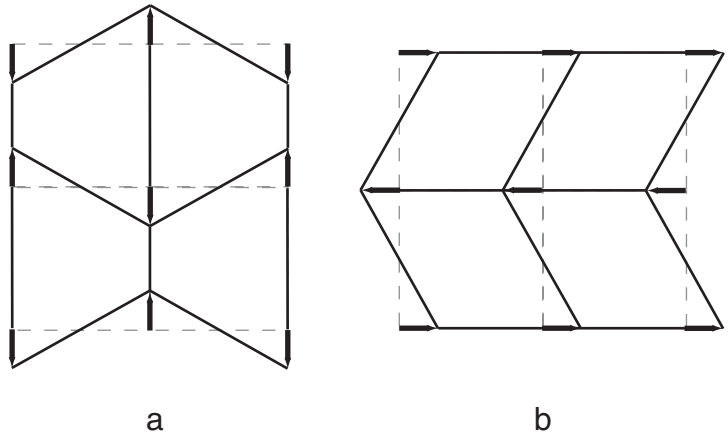


Figure 4.1: Examples of degenerate mesh deformations: (a) the bow-tie pattern; (b) the herringbone pattern

To prevent such deformations from slowly degrading the solution, alternate mesh nodes can be coupled with a small artificial restoring force (Amsden et al., 1980). Ideally, this force should affect flows only at the $2dx$ wavelength level; to guard against degenerate mesh instabilities, but have no influence on the larger flow variations being studied. In SALE, small accelerations are introduced at each vertex, which are based on the surrounding velocity field and tend to keep the vertex velocities from deviating too strongly from adjacent node velocities.

As discussed by Amsden et al. (1980), a fourth-order coupling scheme is effective for the bow-tie mode, but a more diffusive second-order scheme must be used for the herringbone pattern. The fourth-order form used in SALE is given by:

$$u_i^j = u_i^j + \frac{a_{nc}}{4}\left[2\left(u_{i+1}^j + u_i^{j+1} + u_{i-1}^j + u_i^{j-1}\right) - u_{i+1}^{j+1} - u_{i-1}^{j+1} - u_{i-1}^{j-1} - u_{i+1}^{j-1} - 4u_i^j\right] \tag{4.2}$$

in which a_{nc} is a coefficient that governs the amount of coupling. The second-order form is given by:

$$u_i^j = u_i^j + \frac{a_{nc}}{4}\left[\left(u_{i+1}^j + u_i^{j+1} + u_{i-1}^j + u_i^{j-1}\right) - 4u_i^j\right] \tag{4.3}$$

These equations are implemented in SALE in such a way as to enable both order terms to be applied. The coefficient **xi** controls the relative order of the coupling, ranging from 0.0 for second order coupling alone to 1.0 for purely fourth order coupling. Note that the indices in the two equations above correspond to the coordinates of the vertices within the mesh. Although representative values for a_{nc} and **xi** are provided with the SALE manual ($a_{nc} \approx 0.05$, **xi** ≈ 0.3), it is essential that the values chosen are appropriate for the particular problem in hand. This may be achieved by simulating a known phenomenon under conditions similar to the one being studied, and comparing the numerical results with known ones.

As alluded to above, a weakness of the alternate-node coupling scheme just described is that it is independent of cell size and time step. Thus, the velocity diffusion occurs more frequently when the time step is smaller, and over greater distances when the cell size is larger. This leads to the very undesirable trait of different calculation results being produced, for the same problem, when modelled using different cell dimensions or time intervals.

The dependence of the alternate-node coupler on time step and cell size has been eliminated in SALES_2 by multiplying the alternate-node coupler coefficient, a_{nc}, by the ratio dt/dt_{anc}, where dt_{anc} is the maximum time step allowed by the Courant condition on sound signal propagation at the start of the calculation: $dt_{anc} = dt_{cou}(t = 0)$. When $dt = dt_{anc}$, the alternate-node coupler is applied as in the original SALE; however, if the time step is less than dt_{anc} then the amount of coupling is reduced such that the absolute amount of coupling is the same.

To ensure equivalence in the absolute coupling effect for two calculations using different mesh resolutions, an identical time step must be used in each run. If this is the case, the factor dt/dt_{anc}, which is inversely proportional to the cell dimension, will be smaller for the calculation with the lower mesh resolution. Thus, the amount of coupling in the lower-resolution calculation is reduced, compared to that used in simulations with the higher resolution, by the ratio of the cell dimensions. It is important, however, that the chosen time step is that which is appropriate for the highest mesh resolution (smallest cell size) used. This is because the time step restrictions employed by SALES_2 will not allow a large time step, which might be appropriate for a low-resolution simulation, to be used in a higher resolution simulation; that is, dt/dt_{anc} cannot be greater than 1. Furthermore, the user should be aware that the value of a_{nc} defined in the input file will only be applied when the time step is equal to the time step as defined by the Courant condition at the beginning of the calculation: $dt/dt_{anc} = 1$. Consequently, finding the appropriate value for a_{nc} should be carried out using the mesh resolution likely to be used for most of the simulations.

4.2 Equation of state

Before discussing the equation of state currently used in SALES_2 it is worth discussing equations of state more generally.

An equation of state relates three thermodynamic variables, pressure p, density ρ (or specific volume $V = 1/\rho$), and specific internal energy E (or temperature, T). The equation of state is different for different materials and is a complex function of the molecular and atomic structure of a given substance. The response of a material to the shock of an impact is primarily governed by its equation of state.

Geological materials are too complex for any analytical derivation of their equation of state. Therefore, the few methods that have been developed for describing the equation of state of these materials are based on empirical observations. As a result, there is still considerable uncertainty in the equation of state of geological materials in the velocity range at which most impacts occur in the solar system.

Numerical hydrocodes typically compute the material density ρ and the specific internal energy E every cycle, from which the pressure p must be calculated in the next cycle. The most convenient form of the equation of state for hydrocode calculations is thus

$$p = p(\rho, E)$$

One should note that most thermodynamic studies derive equations of state in terms of temperature T, not E. Real physical analysis must use the temperature form and work in this area has lead to ANEOS, a semi-analytical equation of state. However, although one can, in some cases, derive one from the other by fast Newtonian iteration, in general there is only an indirect relationship between temperature and internal energy. Therefore, the most widely used equations of state have been developed with the above form.

4.2.1 The liquid equation of state

The pressure of a liquid subjected to a small compression or expansion is a linear function of the density fluctuation about the uncompressed density ρ_0:

$$p = c_B^2(\rho - \rho_0) \tag{4.4}$$

where $c_B = \sqrt{K_0/\rho_0}$ is the bulk sound speed. Although it appears that this equation has no dependence on E, this is a false impression as K_0 is a weak function of temperature. K_0 is the isentropic bulk modulus for sound waves and the isothermal bulk modulus for slow static compression. These two modulii differ by less than one percent for most substances.

In hydrocode computations, densities much less than ρ_0 will occur during rapid expansion. Physically, such densities indicate that the material in the

computational cell is no longer a continuum, but may be in the form of droplets or solid fragments separated by gas or vacuum instead. Therefore, the pressure computed from this equation of state must be limited to avoid artificially large tensions developing. This is achieved by testing the ratio ρ/ρ_0; if it is less than some amount, often chosen between 0.8 and 0.95, the pressure is set to zero or, in sophisticated equations of state, to the vapour pressure over the liquid or solid.

4.2.2 The perfect gas equation of state

The pressure in a perfect gas is a simple function of density and internal energy:

$$p = (\gamma - 1)\rho E \tag{4.5}$$

where γ is the ratio of the specific heat at a constant pressure to that at constant volume, $\gamma = C_p/C_V$.

4.2.3 The Mie-Grunsien and Murnaghan equation of state

The liquid equation of state (4.4) and the perfect gas equation of state (4.5) are sometimes combined to yield a more realistic equation of state for solids called the "stiffened gas" equation of state:

$$P = c_B^2(\rho - \rho_0) + (\gamma - 1)\rho E \tag{4.6}$$

The Mie-Grunsien and Murnaghan equations of state are a combination of the perfect gas and liquid equations of state and an extension of the liquid equation of state respectively. Although there application in geophysics and planetary science is widespread, a more specific equation of state is required for the problem of impact cratering. For further reading on these two equations of state refer to Appendix II of Melosh (1989).

4.2.4 The Tillotson equation of state

This is the equation of state currently used in SALES_2. It was specifically designed for high-velocity impact computations, in 1962 by J. H. Tillotson. The equation was designed to duplicate the linear shock-particle velocity relation at low pressures and to extrapolate to the Thomas-Fermi limit at high pressures. This equation of state also has parameters that allow it to describe the unloading of shocked material into the vapour phase.

The Tillotson equation has two different forms, depending on whether the material is compressed to higher density than its zero-pressure form or expanded to lower density. The form used in the compressed region ($\rho/\rho_0 \geq 1$) and for cold expanded states where the energy density is less than the energy of incipient vaporization, $E < E_{iv}$, is

$$p = \left[a + \frac{b}{(E/(E_0\eta^2) + 1)}\right] \rho E + A\mu + B\mu^2 \tag{4.7}$$

where $\eta = \rho/\rho_0$, $\mu = \eta - 1$ and a, b, A, B, and E_0 are the Tillotson parameters. In spite of the notation, E_0 is not the initial energy density of the substance: it is merely a parameter that is often close to the vaporization energy. The initial energy density E must actually be zero to ensure that $p = 0$ in the initial state. Note that a low-density pressure cutoff in cold expanded states for $\eta < 0.8$ to 0.95 must be applied, as discussed earlier.

The parameter a is usually chosen to equal 0.5 to give the correct high-pressure limit. Note that at high pressure and temperature the pressure tends towards $a\rho E$, and $a = 0.5$ is not what would be expected from the perfect gas equation of state. However, $a = 0.5$ has been found to fit observational data better.

The linear shock-particle velocity model, derived from observational results from shock-wave experiments, relates the shock-wave velocity U to the particle velocity u_p by:

$$U = C + Su_p$$

The two constants C and S can be expressed in terms of the Tillotson parameters by

$$C = \sqrt{(A/\rho_0)}$$

(A is thus equivalent to K_0) and

$$S = \frac{1}{2}\left[1 + \frac{B}{A} + \frac{(a+b)}{2}\right]$$

In the expanded state, $\eta \leq 1$, when the internal energy exceeds the energy of complete vaporization $E > E_{cv}$, the pressure is given by:

$$p = a\rho E + \left[\frac{b\rho E}{(E/(E_0\eta^2) + 1)} + A\mu e^{-\beta(\rho/\rho_0 - 1)}\right] e^{-\alpha(\rho/\rho_0 - 1)^2} \tag{4.8}$$

where α and β are constants that control the rate of convergence of this equation to the perfect gas law.

In order to make the transition between the two regimes smooth it has been found that in the partial vaporization regime, when $\rho/\rho_0 < 1$ and $E_{iv} < E < E_{cv}$, the pressure is best computed from a hybrid formula:

$$p = \frac{(E - E_{iv})p_h + (E_{cv} - E)p_c}{E_{cv} - E_{iv}} \tag{4.9}$$

where p_h is computed from the expanded (or hot) Tillotson equation and p_c is computed from the compressed Tillotson equation.

When high-pressure phase transformations occur in a shocked solid the internal energy E computed from the Hugoniot equation includes the transition energy E_{tr} from the low- to high- pressure phases. The internal energy

E_{hpp} in the Tillotson equations for the high-pressure phase must be corrected for this transition energy by using the following relation to compute the internal energy

$$E_{hpp} = \frac{p(V_{00} - V)}{2} - E_{tr}$$

where V_{00} is the specific volume of the low-pressure phase. V_{00} is taken to be zero at standard temperature and pressure.

4.2.5 SALES_2 equation of state algorithm

In SALES_2 the equation of state is defined in the subroutine *eos*. Here the subroutine has been modified to represent the Tillotson equation of state, although the algorithm may be simplified to convert the equation of state to that for a perfect gas, stiffened gas or a linear elastic solid. By setting $a = b = B = \alpha = \beta = E_0 = 0$, equations 4.7 and 4.8 reduce to the liquid equation of state (4.4) with $A = K_0$, the bulk modulus. If instead $A = b = B = \alpha = \beta = E_0 = 0$ then equations 4.7 and 4.8 reduce to the perfect gas equation of state (4.5) with $a = (\gamma - 1)$. To get the stiffened gas equation of state (4.6), therefore, set $b = B = \alpha = \beta = E_0 = 0$, $A = K_0$ and $a = (\gamma - 1)$.

The call to the subroutine *eos* passes the density **ro(ij)**, internal energy **sie(ij)**, the damage coefficient **damag(ij)** and the material type **imat(ij)**. *eos* then decides which state the cell is in (expansion or compression) and calculates the component of the pressure (p_c or p_h) using the appropriate equation (4.7 or 4.8). If the material is fractured the compression term of the pressure is modified accordingly (see section 4.3.4). If the density of the cell becomes too low ($\rho/\rho_0 <$ **density_cutoff**) then the pressure is set to zero and a flag is raised (**density_flag**). The appropriate pressure component (p_c and p_h) is then returned unless the cell is in the transition regime. In this case the hybrid formula (4.9) is used. The new value for the pressure in the cell is then returned with the density flag **low_density(ij)** and a flag identifying the equation of state regime used to calculate the pressure (**eos_iddt(ij)** = 'c' for cold or compressed, 'h' for hot or expanded and 't' for transition).

4.3 Constitutive model

The original SALE code used a very basic constitutive model which only permitted the study of one rheological model: Newtonian fluid flow. The constituive model implemented in SALES_2 allows a number of different material rheologies to be modelled. Rheological models combine the concepts of elasticity, plasticity and viscosity. In the aid of understanding, each type of behaviour is conventionally represented by a simple mechanical model. Elasticity is represented by a spring of stiffness k; viscosity η is represented

by a dash-pot; and plasticity is represented by a frictional contact of resistance σ_Y (see figure 4.2). Thus, a rheological model becomes a combination of these simple elements.

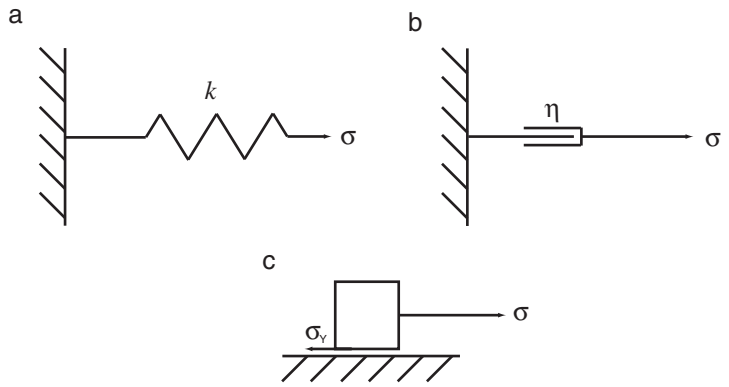


Figure 4.2: Mechanical model analogues for (a) elasticity, (b) viscosity and (c) plasticity.

By defining the input parameters correctly SALES_2 may combine these elements to model three idealised rheologies: a Newtonian fluid; a Kelvin substance; and a Bingham substance.

The constitutive model is applied in *stresd* which stands for stress deviator. The stress tensor σ_{ij} may be separated into the hydrostatic component, p and deviatoric components, Π_{ij}. The definition of the deviatoric stress tensor is as follows:

$$\begin{array}{lcll} \Pi_{ij} & = & \sigma_{ij} - p & \text{for } i = j \\ \Pi_{ij} & = & \sigma_{ij} & \text{for } i \neq j \end{array} \tag{4.10}$$

where the indices refer to the standard tensor notation (not cell number!). The hydrostatic component (pressure) is used in the equation of state to calculate volumetric changes discussed above. The deviatoric component is used in the constitutive model to predict changes in distortion.

The strain tensor ϵ_{ij} may be equivalently separated into a volumetric part e and a deviatoric part ($\epsilon'_{ij} = \epsilon_{ij} - e$ for $i = j$, $\epsilon'_{ij} = \epsilon_{ij}$ for $i \neq j$). e is defined as the volumetric strain, or trace, and is given by the expression (in cylindrical coordinates):

$$e = \frac{1}{3}(\epsilon_{xx} + \epsilon_{yy} + \epsilon_{\theta\theta}) \tag{4.11}$$

Similar expressions exist for the deviatoric strain rate $\dot{\epsilon}'_{ij}$. The deviatoric stresses may be a function of the deviatoric strain (elasticity) or the deviatoric strain rate (viscosity) or both. In SALES_2 the viscous deviatoric stress tensor and the elastic deviatoric stress tensor are calculated separately.

4.3.1 Viscosity and Newtonian fluid flow

Writing the equation for a Newtonian fluid in terms of deviatoric stress and strain rate gives,

$$\Pi_{ij}\Big|_{viscous} = 2\eta\dot{\epsilon}'_{ij} \tag{4.12}$$

where η is the shear viscosity.

In two dimensions the deviatoric strain rate in a cell may be calculated from the velocity gradient across it less the volumetric strain, or trace:

$$\begin{aligned} \dot{\epsilon}'_{xx} &= \frac{\partial u}{\partial x} - \dot{e} \\ \dot{\epsilon}'_{yy} &= \frac{\partial v}{\partial y} - \dot{e} \\ \dot{\epsilon}'_{xy} &= \frac{1}{2}\left(\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\right) \end{aligned} \tag{4.13}$$

If considering three dimensions with axial symmetry then the hoop strain rate $\dot{\epsilon}'_\theta$ is defined as:

$$\dot{\epsilon}'_\theta = \frac{u}{r} - \dot{e} \tag{4.14}$$

Therefore the viscous stresses in cylindrical coordinates may be defined as:

$$\begin{aligned} \Pi_{xx} &= 2\eta\left(\tfrac{\partial u}{\partial x} - \dot{e}\right) + \lambda D \qquad & \Pi_{yy} &= 2\eta\left(\tfrac{\partial v}{\partial y} - \dot{e}\right) + \lambda D \\ \Pi_{xy} &= \eta\left(\tfrac{\partial v}{\partial x} + \tfrac{\partial u}{\partial y}\right) & \Pi_\theta &= \mathbf{cyl}\left(2\eta\left(\tfrac{u}{r} - \dot{e}\right) + \lambda D\right) \end{aligned} \tag{4.15}$$

in which λ is the *bulk* viscosity coefficient and D is the velocity divergence. The coefficient **cyl** is zero for Cartesian coordinates and unity for cylindrical coordinates.

By setting the yield strength at zero temperature (**y0**) to zero, or by setting the shear modulus (**mu**) to zero, the effects of elasticity and strength, which will be discussed next, may be ignored. In this case SALES_2 behaves in the same way as the original SALE, modelling a Newtonian fluid.

4.3.2 Elasticity and the Kelvin Substance

The deviatoric strain may be obtained simply by multiplying the deviatoric strain rate by the time step dt, thus,

$$\begin{aligned} \epsilon'_{xx} &= (\tfrac{\partial u}{\partial x} - \dot{e})dt \qquad & \epsilon'_{yy} &= (\tfrac{\partial v}{\partial y} - \dot{e})dt \\ \epsilon'_{xy} &= \tfrac{1}{2}\left(\tfrac{\partial v}{\partial x} + \tfrac{\partial u}{\partial y}\right)dt & \epsilon'_\theta &= \mathbf{cyl}\left(\tfrac{u}{r} - \dot{e}\right)dt \end{aligned} \tag{4.16}$$

The elastic deviatoric stress tensor may then be calculated from Hooke's law, namely:

$$\Pi_{ij}\Big|_{elastic} = 2\mu\epsilon'_{ij} \tag{4.17}$$

where μ is the shear modulus. Note that, in contrast to the original SALE manual, η represents shear viscosity and μ represents the shear modulus.

In the solution algorithm of SALES_2 the total deviatoric stress tensor is the vector sum of the elastic and viscous contributions:

$$\Pi'_{ij} = 2\mu\epsilon'_{ij} + 2\eta\dot{\epsilon}'_{ij}. \tag{4.18}$$

Thus, the SALES_2 hydrocode models a solid material as a spring and a dash pot in parallel, a rheological model known as a Kelvin substance. If a constant deviatoric stress, S, is applied to a Kelvin substance unstrained at time $t = 0$ then the deviatoric strain at time t is given by:

$$\epsilon' = S/2\mu \left(1 - e^{-\mu t/\eta}\right) \tag{4.19}$$

Thus, if the viscosity η is much less than the stiffness μ the substance behaves essentially elastically (set **visc** << **mu**). For non-zero viscosity, however, the strain will eventually tend to $S/2\mu$. This model is the simplest which corresponds to a solid in which vibrations are damped by internal friction. Consequently, it has frequently been used to model the behaviour of crustal rocks over short time-scales. The simulations of complex crater collapse discussed in subsequent chapters use this rheological model to represent the target under normal pre-impact conditions.

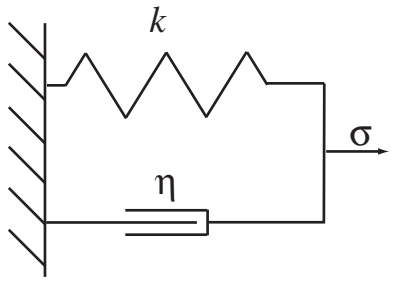


Figure 4.3: Mechanical model of a Kelvin substance

4.3.3 Plasticity, Yield Strength and the Bingham Substance

The Bingham substance (Fig. 4.4) may be represented by a spring connected in series with a frictional block and a dash-pot in parallel. Thus, it has much in common with a plastic material: The Bingham substance behaves as an elastic body for stresses less than the yield stress; above the yield stress the Bingham substance starts to deform permanently. However, the nature of the flow of a Bingham substance differs to that of a plastic in that flow occurs throughout the material; a Bingham substance flows with a viscosity.

The behaviour of a Bingham substance is described by the equations:

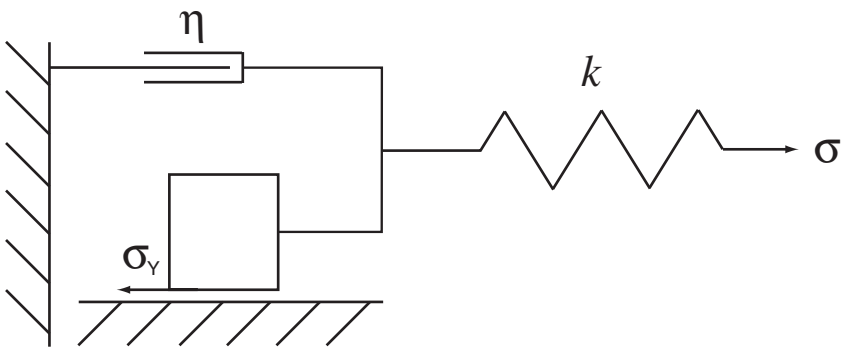


Figure 4.4: Mechanical model of a Bingham substance

$$\sigma_{ij} = 2k\epsilon_{ij}, \text{ if } \sigma < \sigma_Y \tag{4.20}$$

$$\sigma_{ij} = \sigma_Y + 2\eta\dot{\epsilon}_{ij}, \text{ if } \sigma > \sigma_Y. \tag{4.21}$$

To adequately model a Bingham substance, therefore, the constitutive relation in SALES_2 must also incorporate. The approach adopted in SALES_2 to account for plasticity is to reduce the elastic contribution to the stress tensor when the yield strength of the material is exceeded. The yield strength Y_{HEL} at zero temperature is defined in the input file by the parameter **y0**. This is then modified to account for material weakening at increased temperature by the equation:

$$Y_{HEL} = Y_0 \left(1 - \frac{E}{E_m}\right)^2 \tag{4.22}$$

where E is the specific internal energy and E_m is the internal energy required for melting.

Whether or not the yield strength is exceeded in a particular cell is determined by a comparison between the yield strength and an *invariant* measure of the stress in a cell. That is, a function of the stress tensor that is independent of the choice of coordinate axes. The invariant used is the second invariant of the deviatoric stress tensor J_2 defined as:

$$J_2 = \frac{1}{2}\Pi_{ij}\Pi_{ij} = \frac{1}{2}\left((\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_\theta)^2 + (\sigma_\theta - \sigma_1)^2\right) \tag{4.23}$$

where summation over the indices is implied and σ_1, σ_2 and σ_θ are the principal stresses.

If J_2 exceeds the square of the yield strength then the elastic stress component is reduced in relation to how much the yield strength has been exceeded by. The exact amount of stress reduction is given by s_{fac}, where

$$s_{fac} = \sqrt{\frac{Y^2}{J_2}}. \tag{4.24}$$

If J_2 exceeds the square of the yield strength Y_{HEL}^2 then the deviatoric stresses are reduced in relation to how much the yield strength has been exceeded by. The strains are increased accordingly. The exact amount of stress reduction and strain increase is given by s_{fac}, where

$$s_{fac} = \sqrt{\frac{Y_{HEL}^2}{J_2}}. \tag{4.25}$$

The rheological model available to SALES_2 that most closely analogues a Bingham substance is shown in Figure 4.5. In this rheological model the material behaves as a Kelvin substance at stresses below the yield strength. At stresses above the yield stress the material behaves as a Newtonian fluid. Thus, the deviatoric stress for this pseudo-Bingham rheology is given as:

$$\Pi_{ij} = 2\mu\epsilon'_{ij} + 2\eta\dot{\epsilon}'_{ij}, \text{ if } J_2 < Y^2 \tag{4.26}$$

$$\Pi_{ij} = 2\mu\epsilon'_{ij}.s_{fac} + 2\eta\dot{\epsilon}'_{ij}, \text{ if } J_2 > Y^2 \tag{4.27}$$

where η is the effective viscosity of the Bingham substance at stresses below and above the yield stress.

This rheological model may be invoked by setting the yield strength at zero temperature (**y0**) to the Bingham yield strength of the material to be studied.

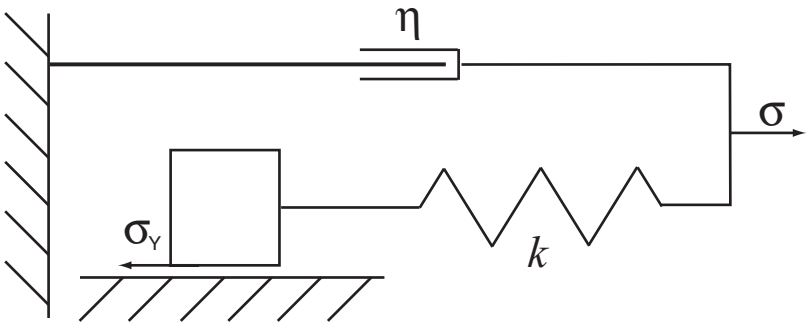


Figure 4.5: Mechanical model of the pseudo-Bingham substance modelled by SALES_2.

4.3.4 Grady-Kipp fragmentation algorithm

Tensile strength is modelled in SALES_2 using the Grady-Kipp model of dynamic fragmentation. What follows is a very brief description of the algorithm. For a far more complete exposition of the physical basis for this model, the implementation into the hydrocode and a comparison between the model and laboratory impacts, the interested reader is referred to Melosh et al. (1992).

In essence, the approach of the fragmentation algorithm is to assess the amount of tensile fracture in a cell and reduce the cohesive strength accordingly. The model has been used to study dynamic fragmentation during impacts, spallation and break-up of meteoroids during atmospheric entry. However, for large scale impact events the algorithm should be used with caution as it does not describe the system of faults and joints typical for natural rocks. In this case any pre-event damage of the target should be treated specifically (Ivanov et al., 1997).

The amount of fracturing is calculated on a cell-by-cell basis. Fracturing in the cell is parameterised by the damage quantity D ($0 \leq D \leq 1$). D is 1.0 for wholesale fracturing in the cell: the material is a collection of unconnected fragments and offers no resistance to further extensional strain. $D = 0.0$ corresponds to totally intact material in the cell. The fragmentation model is implemented in such a way that the amount of damage affects only the cohesive strength of the material. Thus, only the shear modulus is reduced for damaged material; the elastic modulus of completely damaged material is reduced to the bulk modulus. The equation implemented in SALES_2 to account for the reduction in the shear modulus is:

$$\Pi_{ij} = 2\mu(1 - D)\epsilon'_{ij}. \tag{4.28}$$

The fragmentation model assumes that within the continuum material there are points of weakness (cracks or flaws) which grow under tensile stress. The damage is related to the number and size of these flaws by the simple expression:

$$D = nV, \tag{4.29}$$

where n is the number of cracks per unit volume and $V = 4/3\pi l^3$ is the volume of a spherical stress-relieved region surrounding the crack of half length l. This accounts for the reduction in stress in the vicinity of a crack.

The number of flaws activated at any time is given by a two-parameter Weibull distribution:

$$N = k\epsilon^m \tag{4.30}$$

where N is the number of flaws per unit volume activated at or below tensile strain ϵ, and k and m are material constants (defined in the input file by: **pweib** and **cweib**).

Once activated, the cracks are assumed to grow at their maximum speed c_g (another input parameter **cg**) and that the number of flaws that will subsequently activate is reduced by the factor $(1 - D)$. The rate of damage increase is approximated by the expression:

$$\frac{dD^{1/3}}{dt} = \frac{(m+3)}{3}\ \alpha^{1/3}\epsilon^{m/3} \tag{4.31}$$

where $\alpha = 8\pi c_g^3 k/[(m+1)(m+2)(m+3)]$.

Laboratory experiments of fracturing have shown that for slow deformation rates the largest flaws dominate: larger flaws begin to grow under smaller stresses. When a flaw begins to grow, it releases stresses in the vicinity of the crack tending to prevent the growth of other flaws. Consequently, large stresses do not accumulate in the vicinity of the crack and smaller flaws are prevented from growing. The result is that under slow deformation complete fracture occurs at a definite strain or stress, known as the "static" stress σ_{static}.

For high rates of deformation the finite velocity of crack growth allows large stresses to accumulate, activating smaller flaws. Thus, the tensile strength in this regime, known as the dynamic fragmentation regime, is apparently greater than for slow deformation. The transition strain rate $\dot{\epsilon}_*$, at which the dynamic tensile strength exceeds the static strength, is given by:

$$\dot{\epsilon}_* = \frac{(m+4)^{(m+4)/3}}{(m+3)^{(m+3)/3}}\ \alpha \left(\frac{\sigma_{static}}{K}\right)^{(m+3)/3} \tag{4.32}$$

This transition strain rate is a sensitive function of the static stress, which is, in turn, a function of the minimum flaw-activation strain ϵ_{min}:

$$\sigma_{static} = K\epsilon_{min} \tag{4.33}$$

where K is the bulk modulus.

For a finite body the minimum flaw-activation strain depends on the volume of the body V as $k\epsilon_{min} = 1/V$, which, when applied to a spherical body of radius R yields:

$$\epsilon_{min}(R) = \left(\frac{4}{3}\pi k R^3\right)^{-1/m}. \tag{4.34}$$

For most materials $m \sim 10$ implying a strong inverse dependence of static stress on the size of the body. This factor is accounted for in the subroutine *celset*.

The fragmentation algorithm in SALES_2 also records the time that fracturing is initiated (**tfrac**), the size of fragments (**fragsz**) and the orientation of the principal axes at failure (**failur**, in cylindrical symmetry). At the time of termination, this information is written to the file: "damage.ouput", from the subroutine *autpsy*. The subroutine *autpsy* also calls a routine *frstat* to analyse the size-frequency distribution of any fragments produced during the calculation. The results of this analysis are output to the file: "fragsz.output".

Chapter 5

Description of parameters

5.1 Input quantities

Start-up and output information		
jname	problem title	character∗80
ibegcont	problem type flag (0=new, 1=cont., 2=restart)	integer
ndump	dump file number for restart	integer
iidlp	IDL tracer output flag (1=on, 0=off)	integer
dtidlp	IDL tracer output time interval	double
ncells	number of tracer cells/vertices	integer
nplots	number of parameters to be recorded	integer
i_cell, j_cell	cells to be monitored	integer(ncells)
plot_type	parameters to be recorded	integer(nplots)
dt0	first time step	double
dtmax	maximum time step (see SALE manual)	double
twfin	problem finish time	double
dtplot	graphics plot interval	double
ilprt	complete printout flag (0=off, 1=on)	integer
dtlprt	complete printout time interval	double
idump	file dump flag (0=off, 1=on)	integer
dtdump	file dump time interval	double

Mesh properties		
gx, gy	gravitational acceleration components (in m/s^2)	double
nx, ny	number of cells in x and y direction	integer
nproj	number of cells in projectile	integer
cyl	cylindrical symmetry flag (1.=cylindrical, 0.=plane)	double
sphgrd	spherical grid extension flag (0=off, 1=on)	integer
dx, dy	cell dimensions (in metres)	double
wb, wl, wr, wt	boundary condition flags (see SALE manual)	integer

Computational information (see SALE manual)		
imp	implicit pressure calculation flag (0=exp., 1=imp.)	integer
inc	incompressible flag (0=compressible, 1=inc.)	integer
irez	rezone flag (0=eulerian, 1=lagrangian 2+=other)	integer
om	relaxation coefficient used in pressure iteration	double
peps	pressure fraction scaling the relaxation factor (rdsdp)	double
eps	allowed relative error in the pressure iteration	double
rf	relaxation factor for the continuous grid rezoning	double
artvis	artificial (bulk) viscosity coefficient	double
anc	alternate node-coupler coefficient	double
xi	1.=4th-order node-coupling, 0.=2nd-order node-coupling	double
a0, b0	convective fluxing parameters	double

Material properties		
numat	number of materials	integer
roi	initial density (Kg/m^3)	double(numat)
siei	initial internal energy (J/Kg)	double(numat)
csound	speed of sound (m/s)	double(numat)
Tillotson equation of state paramters		
ron	density of the reference state ρ_0 (Kg/m^3)	double(numat)
tilla	Tillotson a parameter	double(numat)
tillb	Tillotson b parameter ($a + b = \gamma$)	double(numat)
atill	A - linear bulk modulus (Pa)	double(numat)
btill	B - coefficient of nonlinear elastic term (Pa)	double(numat)
ezero	E_0 - parameter categorizing energy dependence (J/Kg)	double(numat)
alfa	α - constant controlling convergence to perfect gas eos	double(numat)
beta	β - constant controlling convergence to perfect gas eos	double(numat)
eiv	E_{iv} - Specific energy of incipient vaporisation (J/Kg)	double(numat)
ecv	E_{cv} - Specific energy of complete vaporisation (J/Kg)	double(numat)
Constitutive model parameters		
visc	linear shear viscosity (Pa s)	double(numat)
lambda	linear bulk viscosity (Pa s)	double(numat)
mu	elastic shear modulus (Pa)	double(numat)
y0	hugoniot elastic limit at zero temperature (Pa)	double(numat)
siem	specific internal energy at melting (J/Kg)	double(numat)
Fragmentation model parameters		
cg	crack velocity (m/s)	double(numat)
pweib	weibull fracture exponent	double(numat)
cweib	weibull coefficient k (flaws/m^3)	double(numat)

Specified inflow conditions		
roin	density of inflow material (kg/m^3)	double(4)
siein	internal energy of inflow material (J/kg)	double(4)
pap	pressure at boundary (Pa)	double(4)
uin, vin	velocity of inflow material (m/s)	double(4)

5.2 Derived and scalar quantities in the common block

Quantities derived from the input parameters		
acoef	coefficient for GK damage computation	double(numat)
dpcof	peps*roi/(2*(1/(dx*dx) + 1/(dy*dy)))	double(numat)
fcoef	coefficient for the GK fragment size computation	double(numat)
gkfrac	GK fragmentation flag T=on, F=off	logical(numat)
elastc	elasticity flag T=on, F=off (mu=0)	logical(numat)
omcyl	1 - cyl	double
anco	anc/4	double
xicof	(1+xi)/2	double
grfac	1.e6/(nx*ny)	double
twplot	problem time for next plot	double
twlprt	problem time for next printout	double
twidlp	problem time for next IDL tracer record	double
twdump	problem time for next file dump	double
nxp	nx+1 (number of x-vertices)	integer
nyp	ny+1 (number of y-vertices)	integer
ifirst	=1 if wl=1, 2, 3 or 4; =2 if wl=5 or 6	integer
jfirst	=1 if wb=1, 2, 3 or 4; =2 if wb=5 or 6	integer
ilast	=nx if wr=1, 2, 3 or 4; =nx-1 if wr=5 or 6	integer
jlast	=ny if wt=1, 2, 3 or 4; =ny-1 if wt=5 or 6	integer
ifph1	=1 if wl=1, 2, 3, 4 or 6; =2 if wl =5	integer
jfph1	=1 if wb=1, 2, 3, 4 or 6; =2 if wb =5	integer
ilph1	=nx if wr=1, 2, 3, 4 or 6; =nx-1 if wr =5	integer
jlph1	=ny if wt=1, 2, 3, 4 or 6; =ny-1 if wt =5	integer
ilastv	ilast+1	integer
jlastv	jlast+1	integer
ilastm	ilast-1	integer
jlastm	jlast-1	integer

Scalar quantities		
jnm	optional identifier for code version	character*12
iddt	symbol identifying the restriction controlling the timestep	character*1
	g=growing, c=convection, v=viscosity, m=maximum	
t	problem time (s)	double
dt	time step (s)	double
dtmin	minimum time step permitted (dt of cycle 1*1.d-10)	double
ncyc	cycle (time step) counter	integer
ndump	file dump counter	integer
ipres	=1 during iteration to let continuative boundary	integer
maxit	maximum no. of iterations before dt cut in half	integer
numit	no. of iterations reqd. for pressure iteration convergence	integer
loops	no. of dt cuts for non-convergence for a given cycle	integer
loopmx	limits no. of times that dt can be cut	integer
pmax	maximum cell pressure (Pa)	double
vmax	maximum vertex velocity (m/s)	double
drou	scaling factor for the velocity vector plot	double
dtf	time-step factor for convective stability and accuracy	double
dtfrac	time-step factor for Grady-Kipp fragmentation algorithm	double
nofrac	fragmentation flag (T=no damage, F=damage)	logical
grind	central processor time per cell per cycle (milli-secs)	double
rtime	time of last call to newcyc (for use calculating grind)	double
d1	Month/day/year when the job began	character*24
xl	x-coordinate of the leftmost vertex	double
xr	x-coordinate of the rightmost vertex	double
yb	y-coordinate of the bottommost vertex	double
yt	y-coordinate of the topmost vertex	double
third	1/3	double
twlfth	1/12	double

5.3 Cell and vertex variables

Cell variables		
mc	cell mass (kg)	double(node)
ro	density (kg/m^3)	double(node)
vol	volume (m^3)	double(node)
p	pressure (Pa)	double(node)
sie	specific internal energy (J/kg)	double(node)
imat	material type identifier	integer
q	artificial viscous pressure (Pa)	double(node)
eexx	elastic strain	double(node)
eeyy	elastic strain	double(node)
eexy	elastic strain	double(node)
eeth	elastic strain	double(node)
pixx	stress deviator term	double(node)
piyy	stress deviator term	double(node)
pixy	stress deviator term	double(node)
pith	stress deviator term	double(node)
damag	damage due to crack propagation	double(node)
fragsz	size of Grady-Kipp fragments in failed regions	double(node)
tfrac	time at which cell failed (damag=1.)	double(node)
nint	number of cycles between D=0.0 and D=1.0	integer
failur	orientation of principal axes at failure in cyl. sym.	character*1
stmin	initial strain for Grady-Kipp fragmentation algorithm	double(node)
ptmax	maximum cell pressure during run	double(node)
smax	maximum principal compression attained in cell	double(node)
smin	maximum principal tension attained in cell	double(node)
sigma1	maximum principal stress	double(node)
sigma2	intermediate principal stress	double(node)
sigma3	minimum principal stress	double(node)
umom, vmom	cell momentum components	double(node)
mp	intermediate storage for new mc in eulerian calculation	double(node)
siep	intermediate storage for new sie in eulerian calculation	double(node)
umomp, vmomp	intermediate storage for new momentum components	double(node)
rol	lagrangian density (updated in pressure iteration)	double(node)
pl	lagrangian pressure (updated in pressure iteration)	double(node)
rdsdp	reciprocal of numerical derivative for pressure iteration	double(node)
d	velocity divergence	double(node)
rrsum	1/(r1+r2+r3+r4) of the cell	double(node)
nctsie	counter for negative cell energies	integer

Vertex variables		
x	cartesian coordinate in radial direction	double(node)
r	radial coordinate (r=1.0 for cartesian, r=x for cylindrical)	double(node)
y	cartesian coordinate in the axial direction	double(node)
u	x-direction vertex velocity	double(node)
v	y-direction vertex velocity	double(node)
mv	vertex mass	double(node)
rmv	reciprocal of vertex mass	double(node)
ul	lagrangian u	double(node)
vl	lagrangian v	double(node)
ug, vg	grid velocity	double(node)
urel, vrel	relative velocity between grid and material (grid-lagrangian)	double(node)
mvp	intermediate storage for new mv in eulerian calculation	double(node)

Chapter 6

Compilation, running and a case example

6.1 Compilation issues and portability

As mentioned in the introduction, SALES_2 has been written almost exclusively on a Sun-Unix platform and compiled using SUN F77. This compiler has the disadvantage of hiding a multitude of sins that more rigorous compilers may complain about. Such issues have lead to changes to version 2.0 of SALES_2: **please ensure that you have downloaded version 2.1** before attempting to compile on a different platform. If you encounter any compiler errors please report them to the author (email: gareth@lpl.arizona.edu).

SALES_2 **requires the PGPLOT graphics libraries** which you will need to install and then link with the SALES_2 object file in order to make the executable. For more information about PGPLOT, and the libraries themselves, the reader is referred to: http://www.astro.caltech.edu/~tjp/pgplot/.

PGPLOT is available for multiple different compiler-platform combinations; however, it is worth making sure that PGPLOT is successfully installed and operational before attempting to compile SALES_2.

6.2 Example calculation

The input file supplied with the code is set up to illustrate some of the capabilities of the SALES_2 hydrocode. The problem considers a collapsing cliff face. The cliff consists of a rigid layer supported by three fluid layers.

In the example problem the mesh is a 20 x 20 grid made up of 100m x 100m cells. Gravity is in the downward direction (negative). Cylindrical symmetry option is turned off, **cyl**=0.0. To mimic the situation near a cliff face the boundary conditions in this problem are set such that the left wall has no slip allowed, the bottom boundary allows free-slip and the right and top are free lagrangian surfaces. For more information on boundary conditions in SALES_2, please refer to the SALE manual.

In terms of material properties the mesh in the example calculation is

divided into 4 layers. The bottom three layers have identical material properties. The top layer differs only in its value of density (**roi**) and **y0**. As explained in section 4.3 this parameter is the Hugoniot elastic limit for the material at zero temperature. By setting this parameter to a high value (the crushing strength of competent rock) for the fourth layer and zero for the other three layers the upper layer is defined as rigid and the lower layers fluid (as though they have failed under the weight of the upper layer).

For the example problem the fragmentation algorithm is disabled. For information regarding the use of this algorithm the user is referred to Melosh et al. (1992).

The accompanying pdf document "example.pdf" and output file "example.out" are for the example calculation described above and are included for aiding in code verification at other installations.

Bibliography

Amsden, A. A., Ruppel, H. M., and Hirt, C. W., SALE: Simplified ALE Computer Program for Fluid Flow at all Speeds: Technical Report LA-8095, Los Alamos National Laboratory, Los Alamos, NM, 1980.

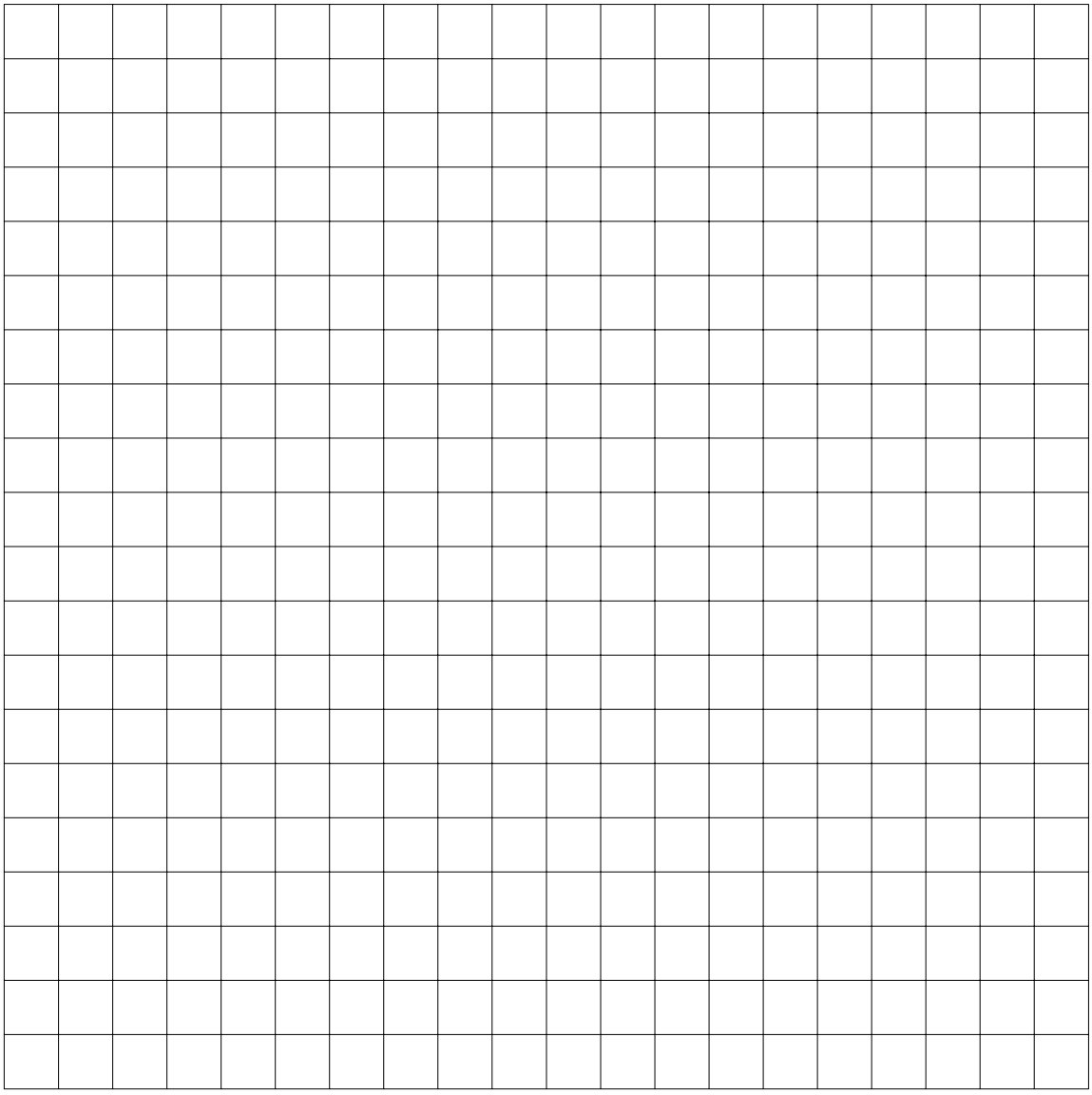
Anderson, C. E., 1987, An overview of the theory of Hydrocodes: International Journal of Impact Engineering, **5**, 33–59.

Ivanov, B. A., Deniem, D., and Neukum, G., 1997, Implementation of dynamic strength models into 2D hydrocodes: Applications for atmospheric breakup and impact cratering.: Int. J. Impact Engineering, **20**, 411–430.

Melosh, H. J., Ryan, E. V., and Asphaug, E., September 1992, Dynamic Fragmentation in Impacts: Hydrocode Simulation of Laboratory Impacts: JGR, **97**, No. E9, 14735–14759.

Melosh, H. J., 1989, Impact cratering: A geological process.:, number 11 in Oxford monographs on geology and geophysics Oxford University Press.

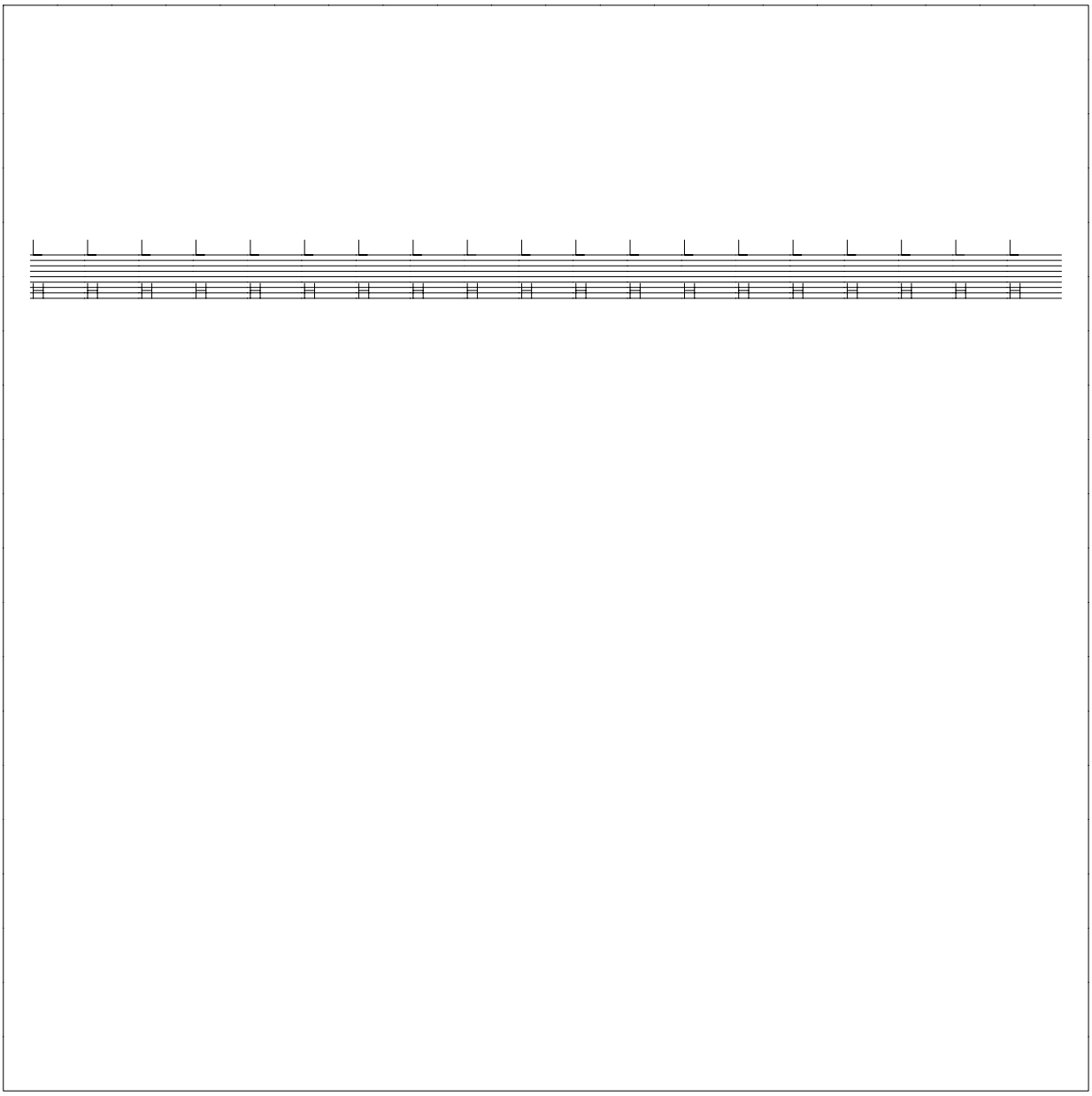
zone plot



t= 0.00000E+00 cycle 0

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

contours for density

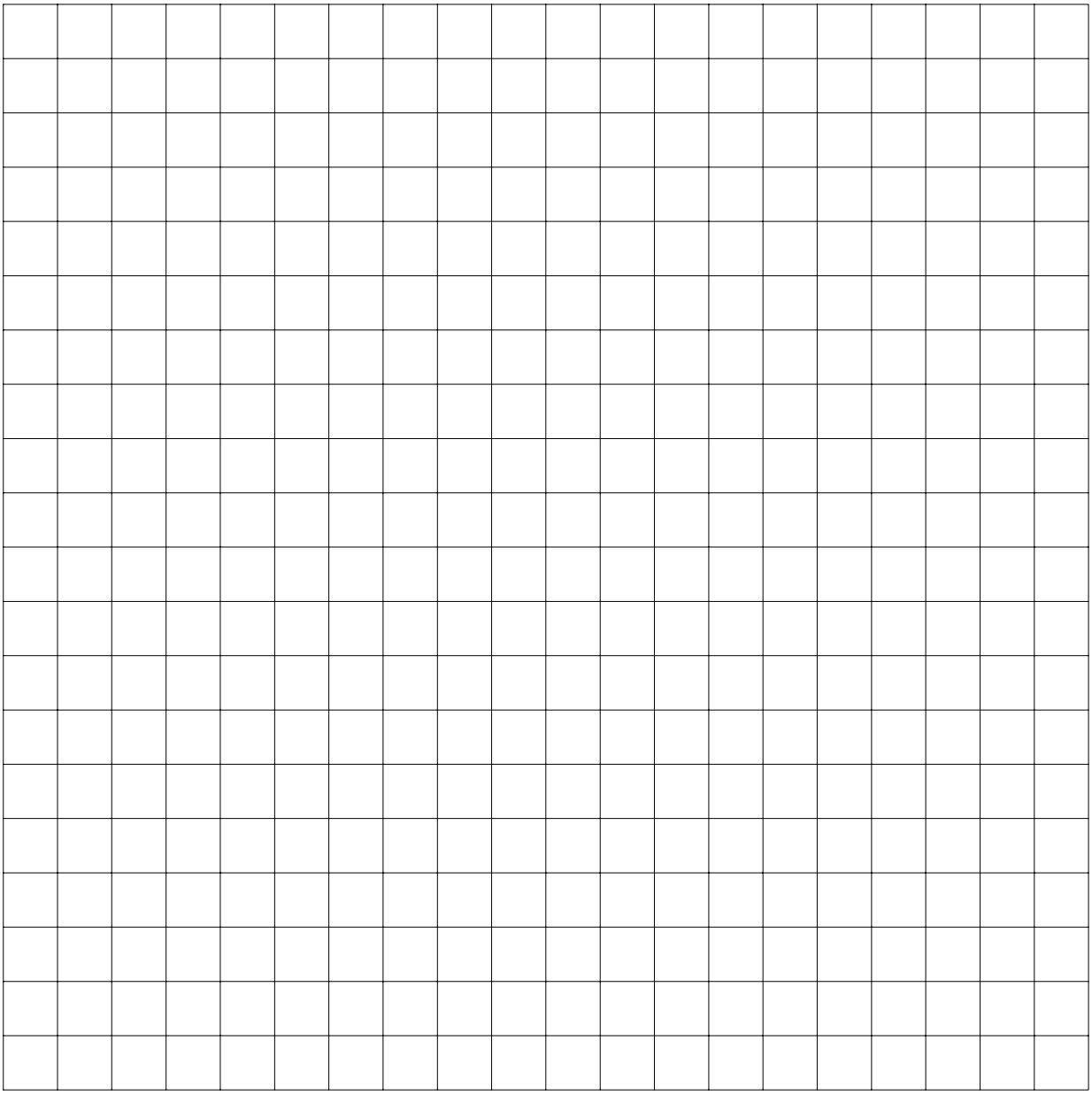


t= 0.00000E+00 cycle 0

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

density min= 2.90000E+03 max= 3.00000E+03 L= 2.91000E+03 H= 2.99000E+03 dq= 1.00000E+01

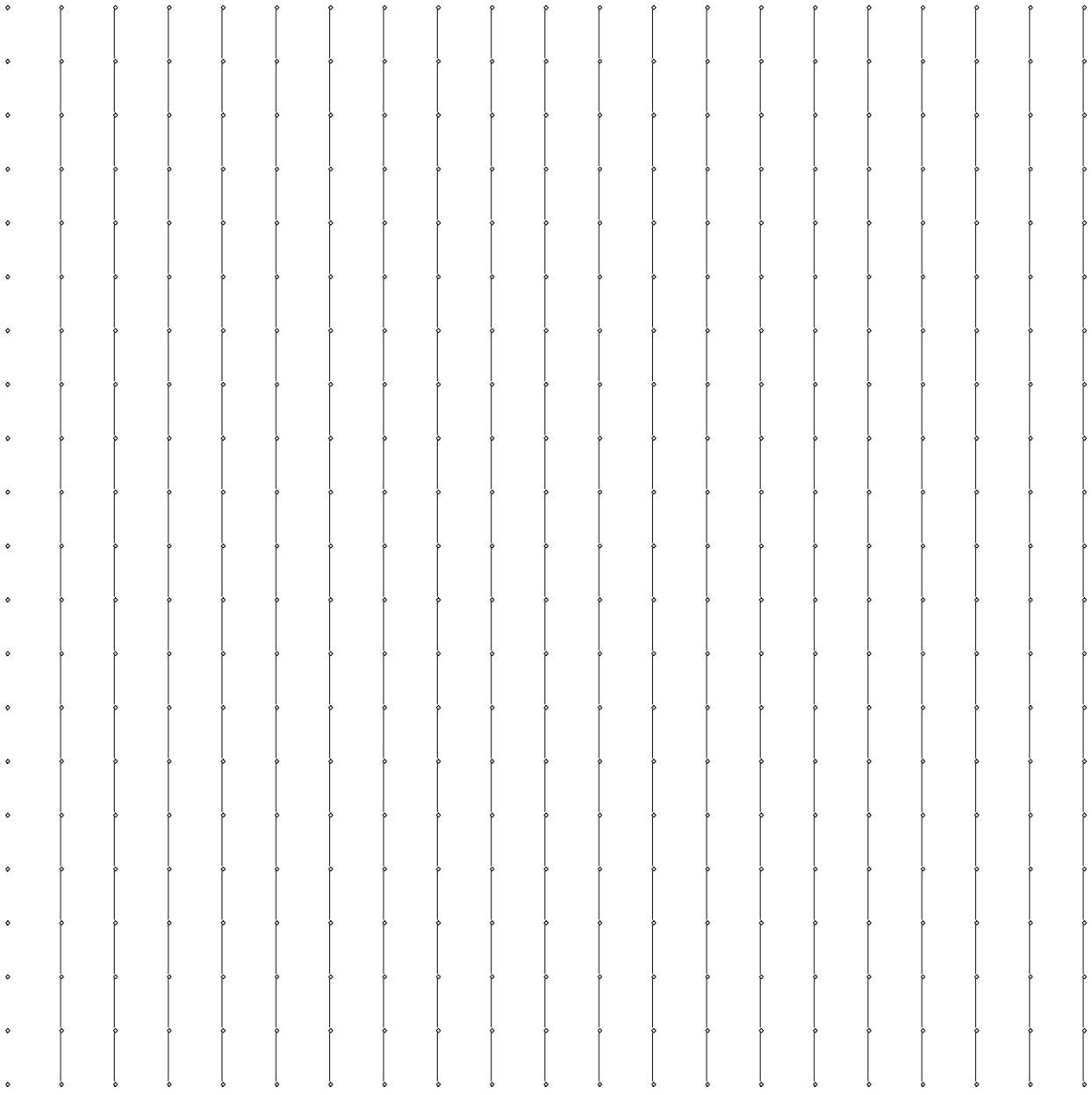
zone plot



t= 1.00000E−05 cycle 1

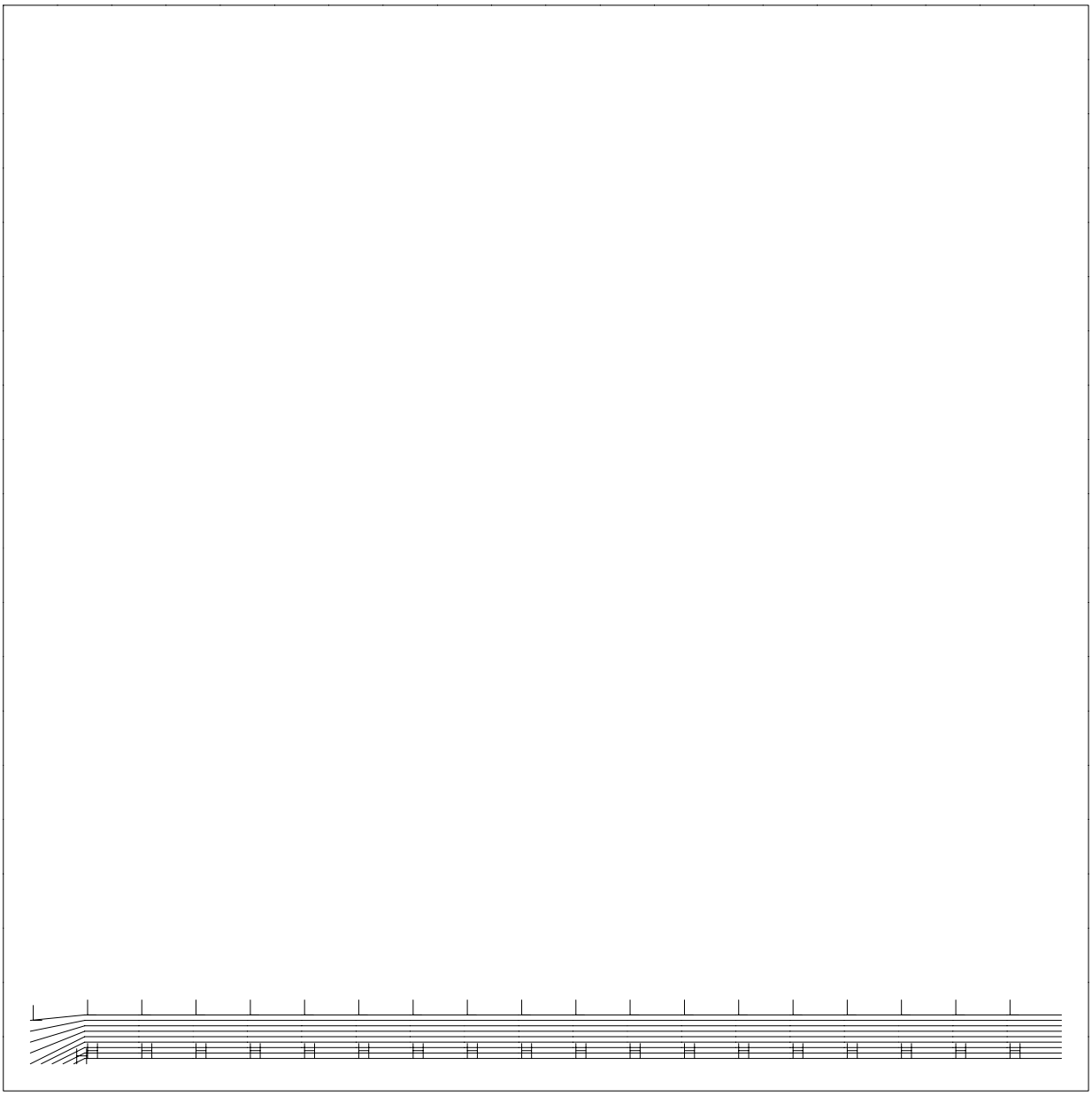
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

velplot



t= 1.00000E−05 cycle 1 vmax= 1.00000E−04
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

contours for pressure

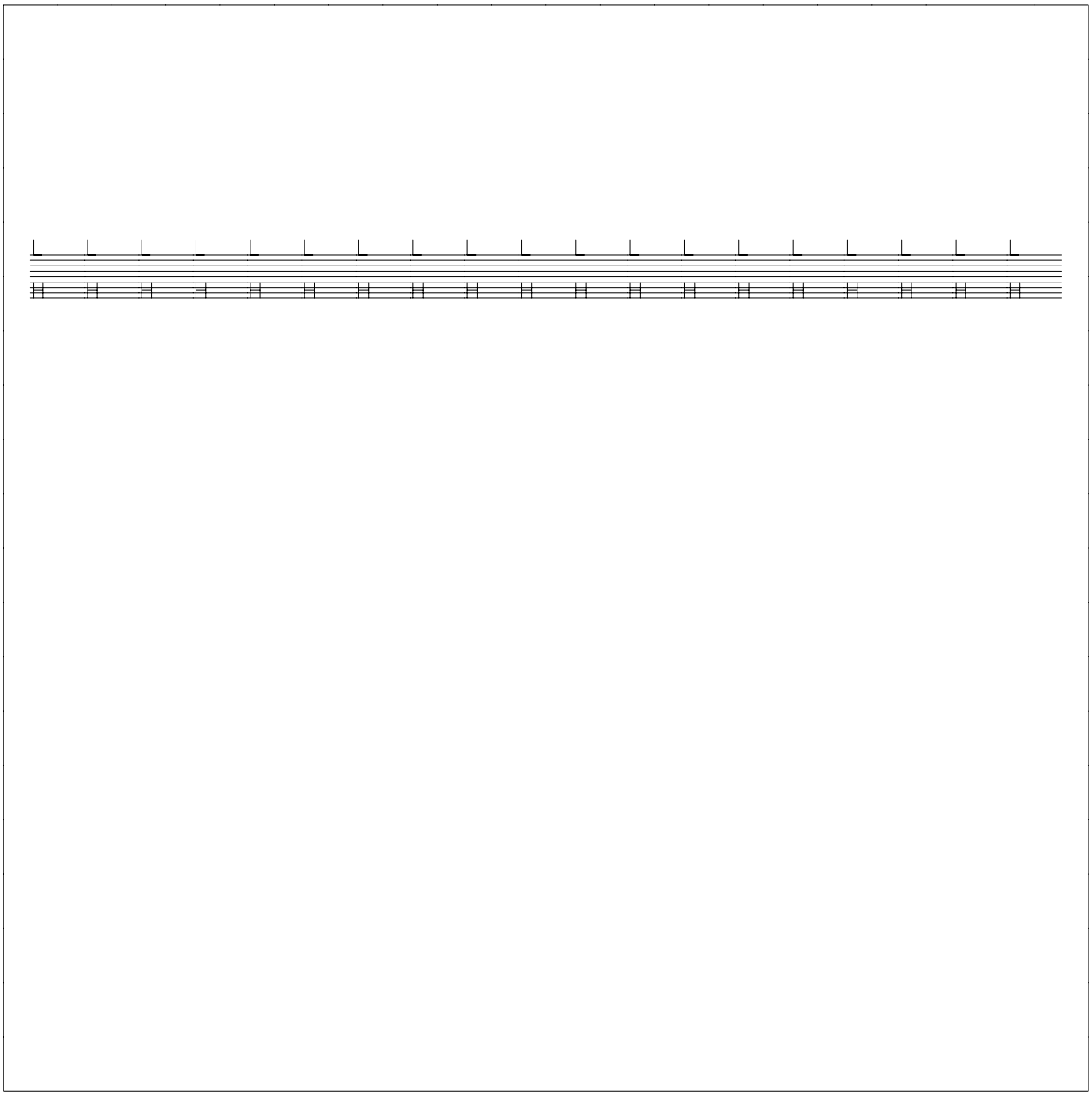


t= 1.00000E−05 cycle 1

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

pressure min=−5.88418E−06 max= 5.30000E−01 L= 5.29947E−02 H= 4.76999E−01 dq= 5.30006E−02

contours for density

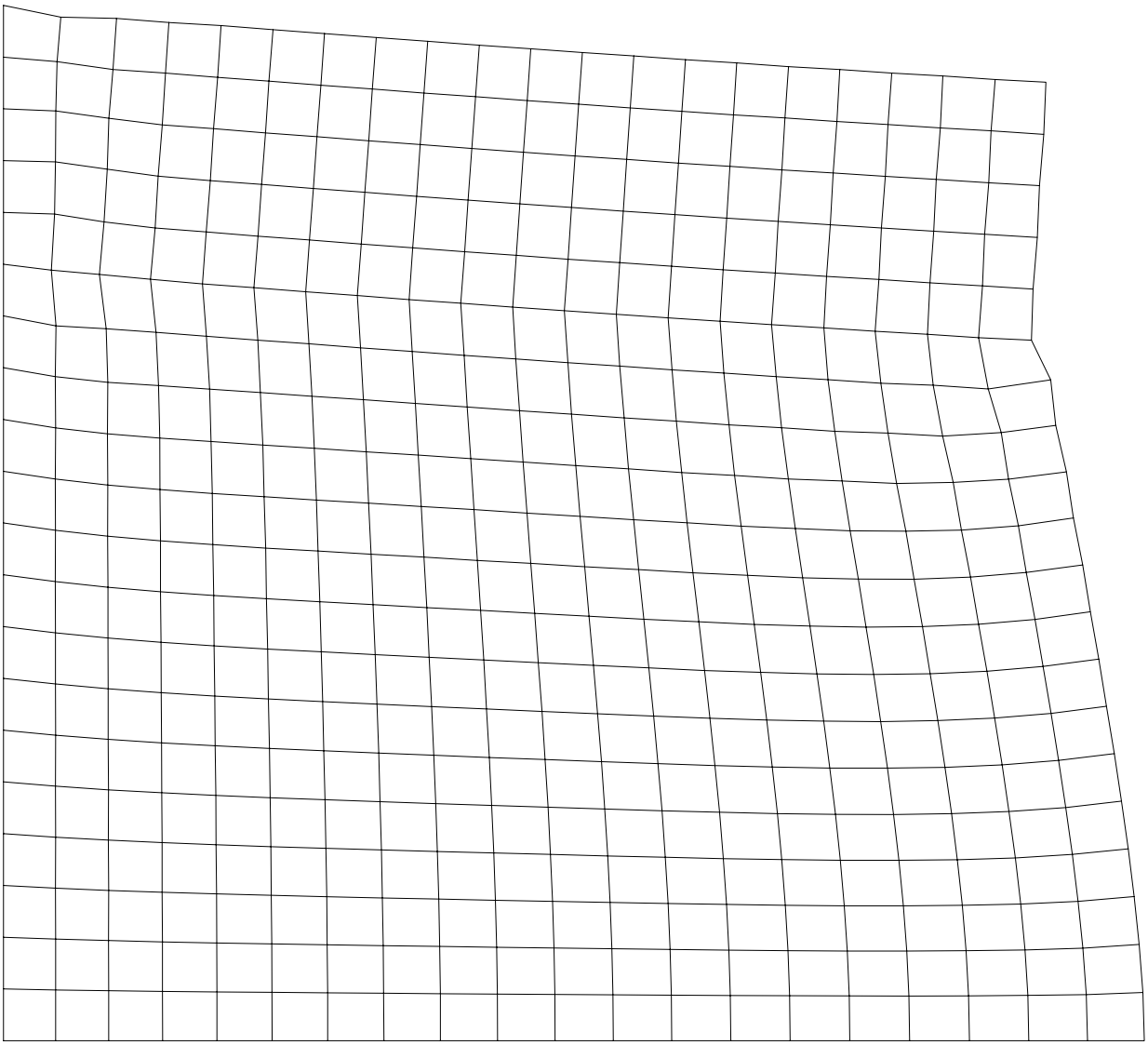


t= 1.00000E−05 cycle 1

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

density min= 2.90000E+03 max= 3.00000E+03 L= 2.91000E+03 H= 2.99000E+03 dq= 1.00000E+01

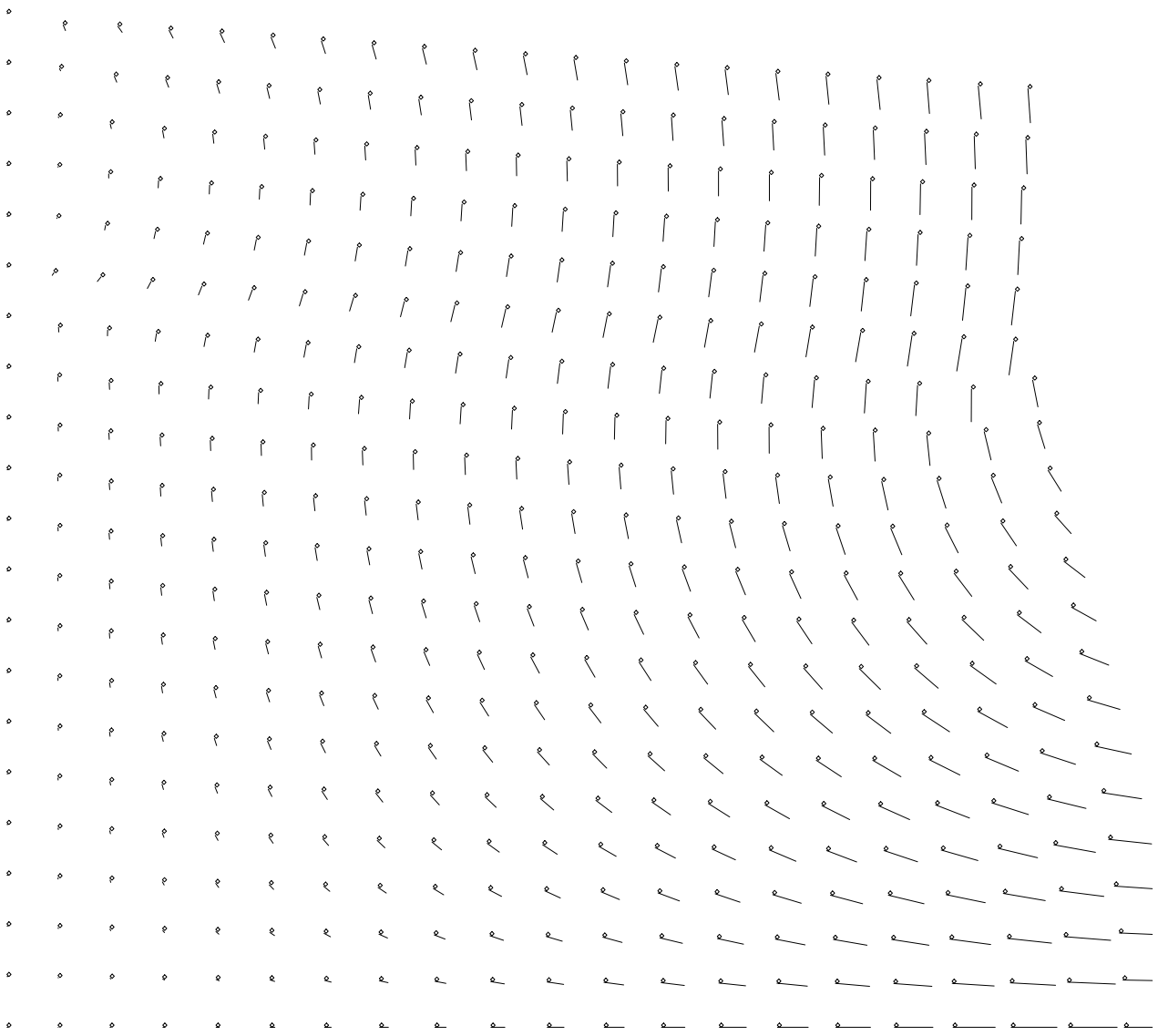
zone plot



t= 1.00021E+01 cycle 3431

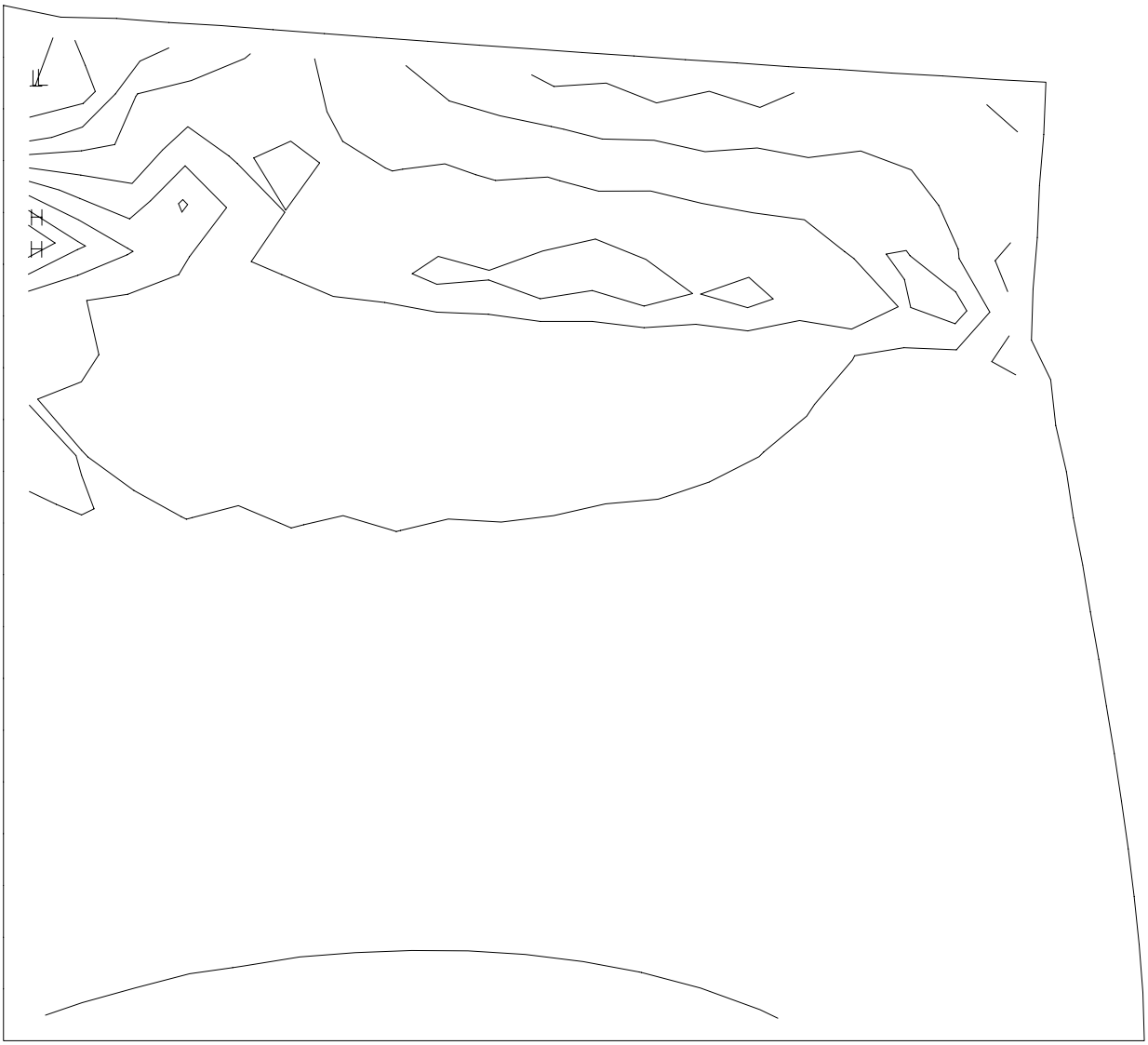
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

velplot



t= 1.00021E+01 cycle 3431 vmax= 2.43666E+01
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

contours for pressure

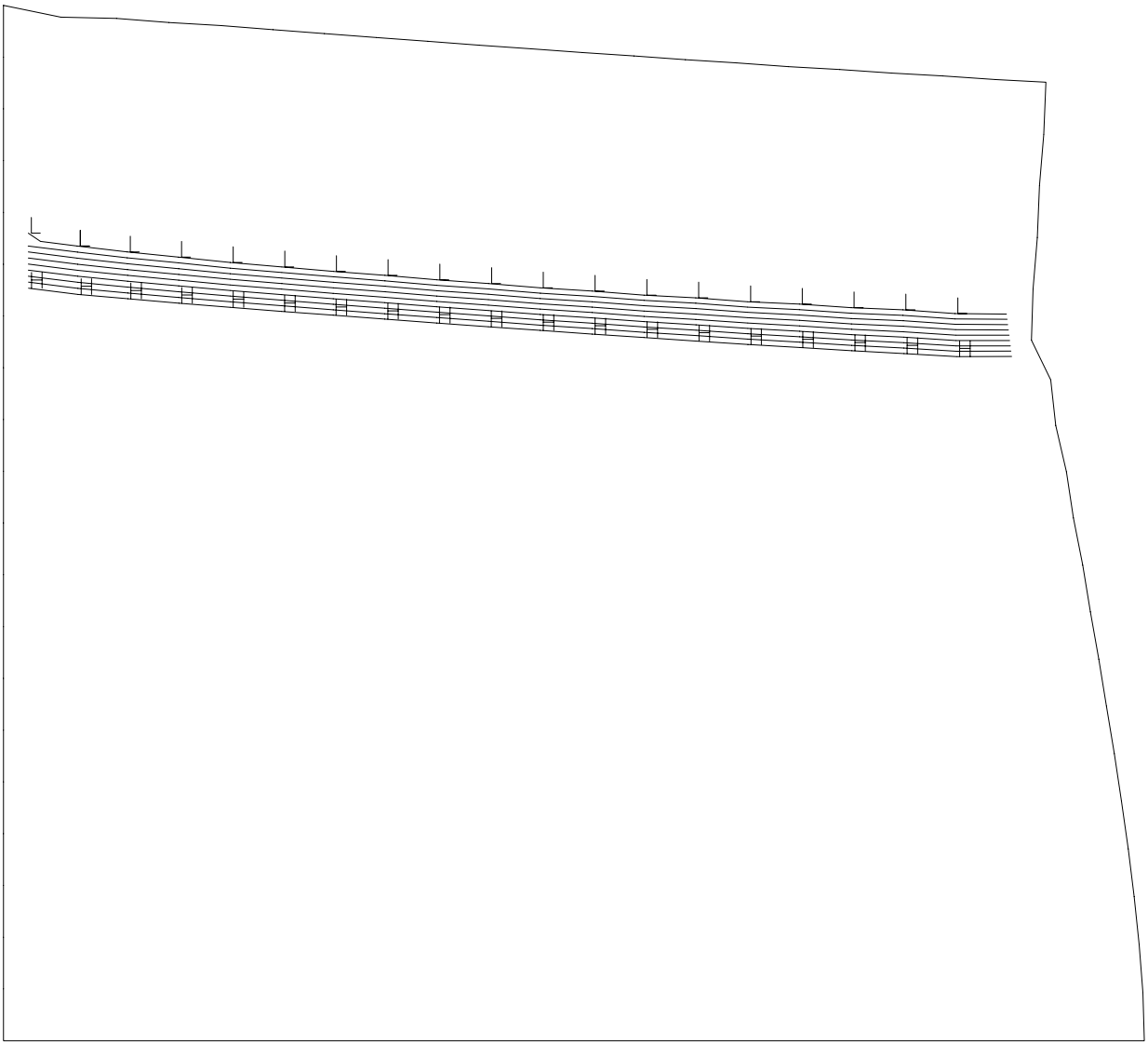


t= 1.00021E+01 cycle 3431

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

pressure min=−1.15542E+08 max= 8.77972E+07 L=−9.52081E+07 H= 6.74632E+07 dq= 2.03339E+07

contours for density

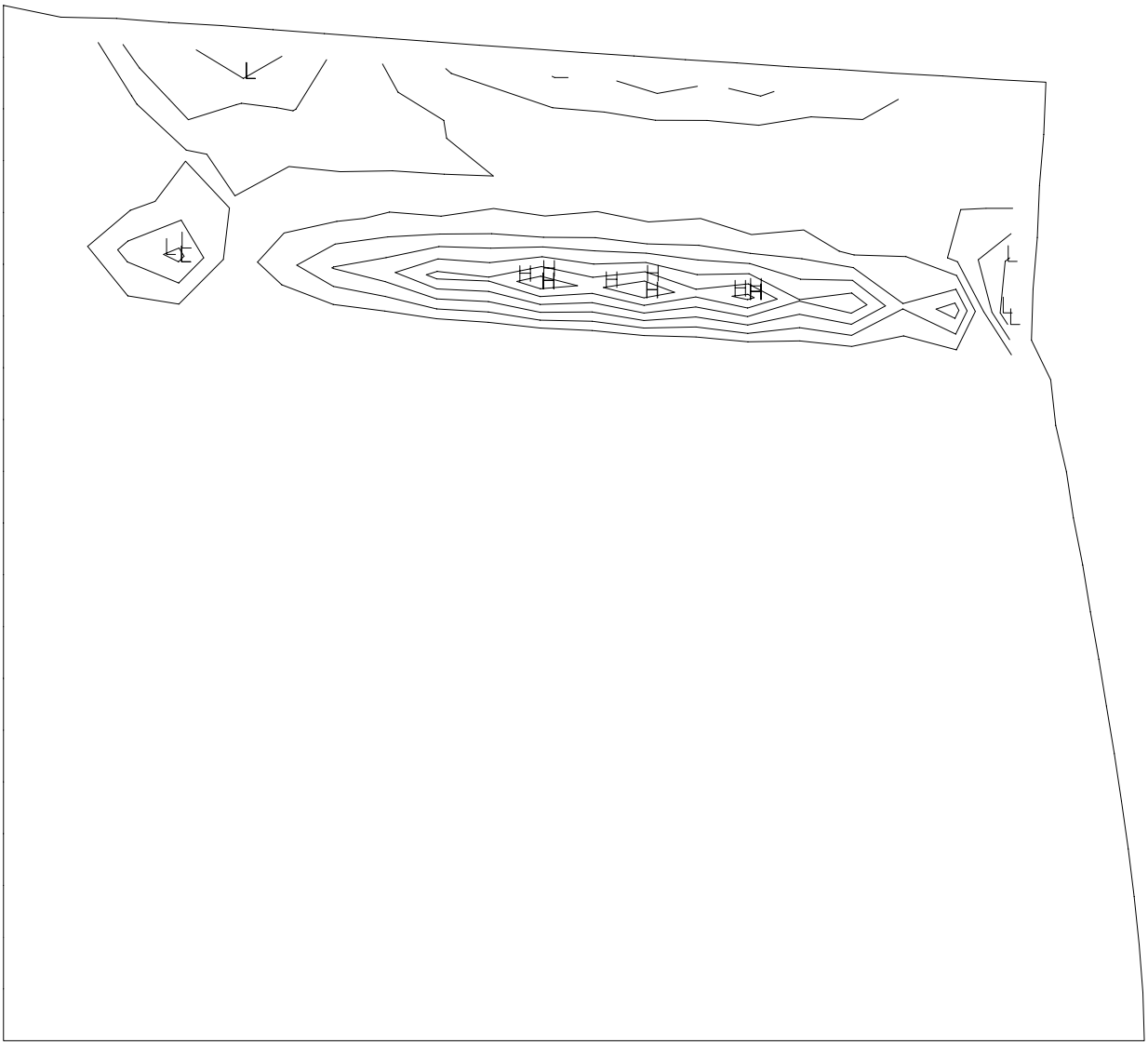


t= 1.00021E+01 cycle 3431

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

density min= 2.89334E+03 max= 3.00172E+03 L= 2.90418E+03 H= 2.99088E+03 dq= 1.08382E+01

contours for int.energy

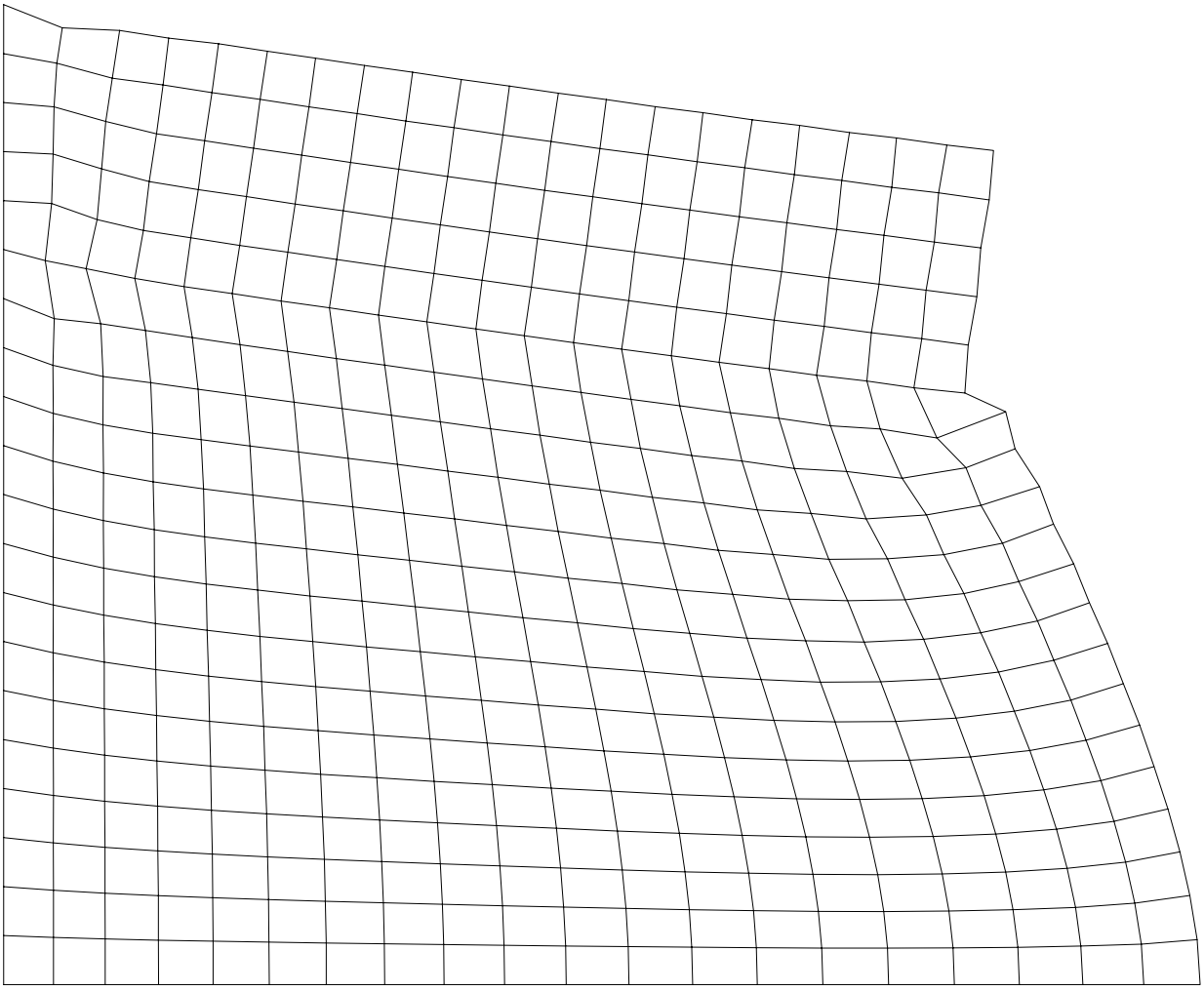


t= 1.00021E+01 cycle 3431

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

int.energy min=−1.64633E+04 max= 3.69400E+04 L=−1.11230E+04 H= 3.15997E+04 dq= 5.34034E+03

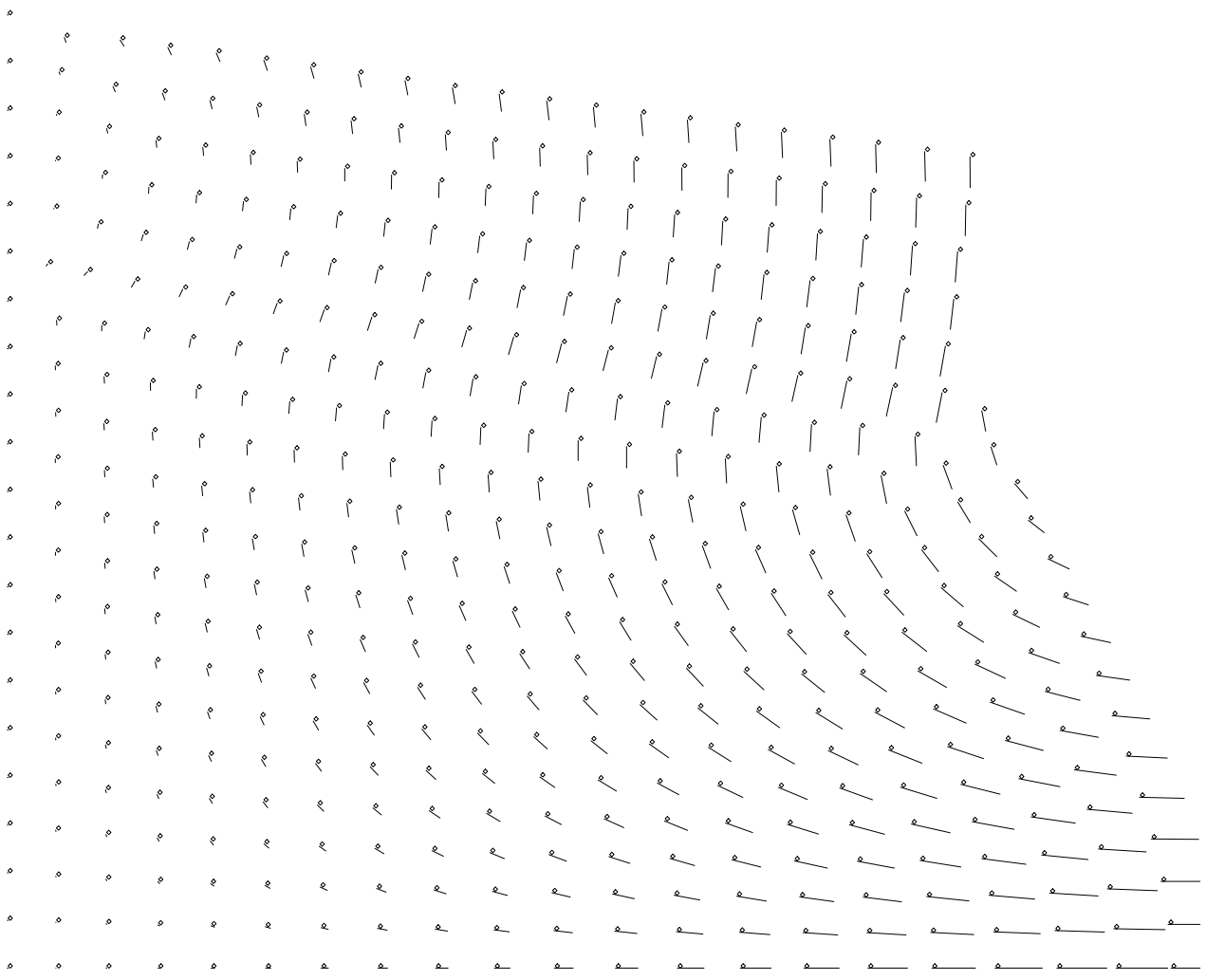
zone plot



t= 2.00003E+01 cycle 6819

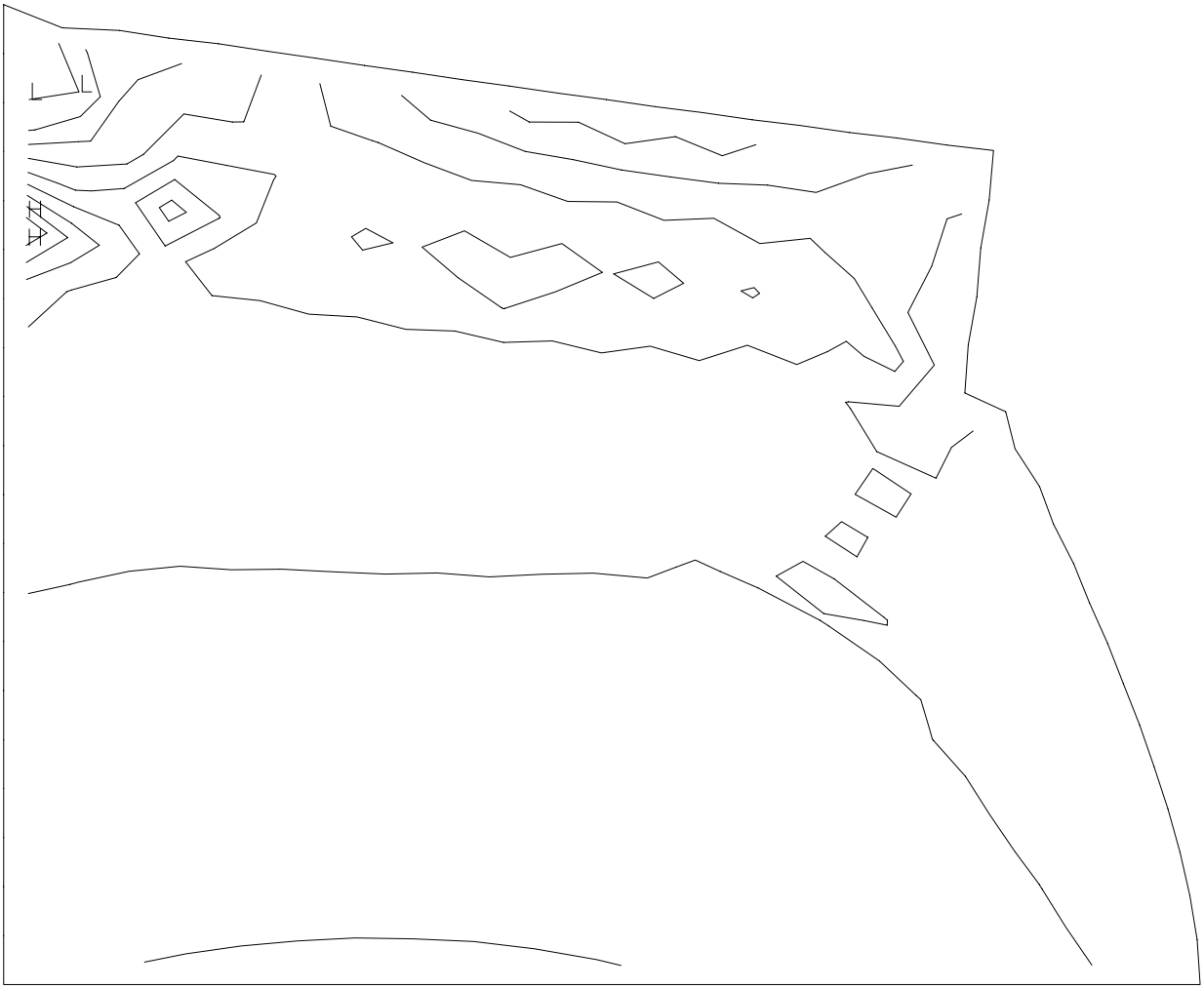
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

velplot



t= 2.00003E+01 cycle 6819 vmax= 2.28753E+01
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

contours for pressure

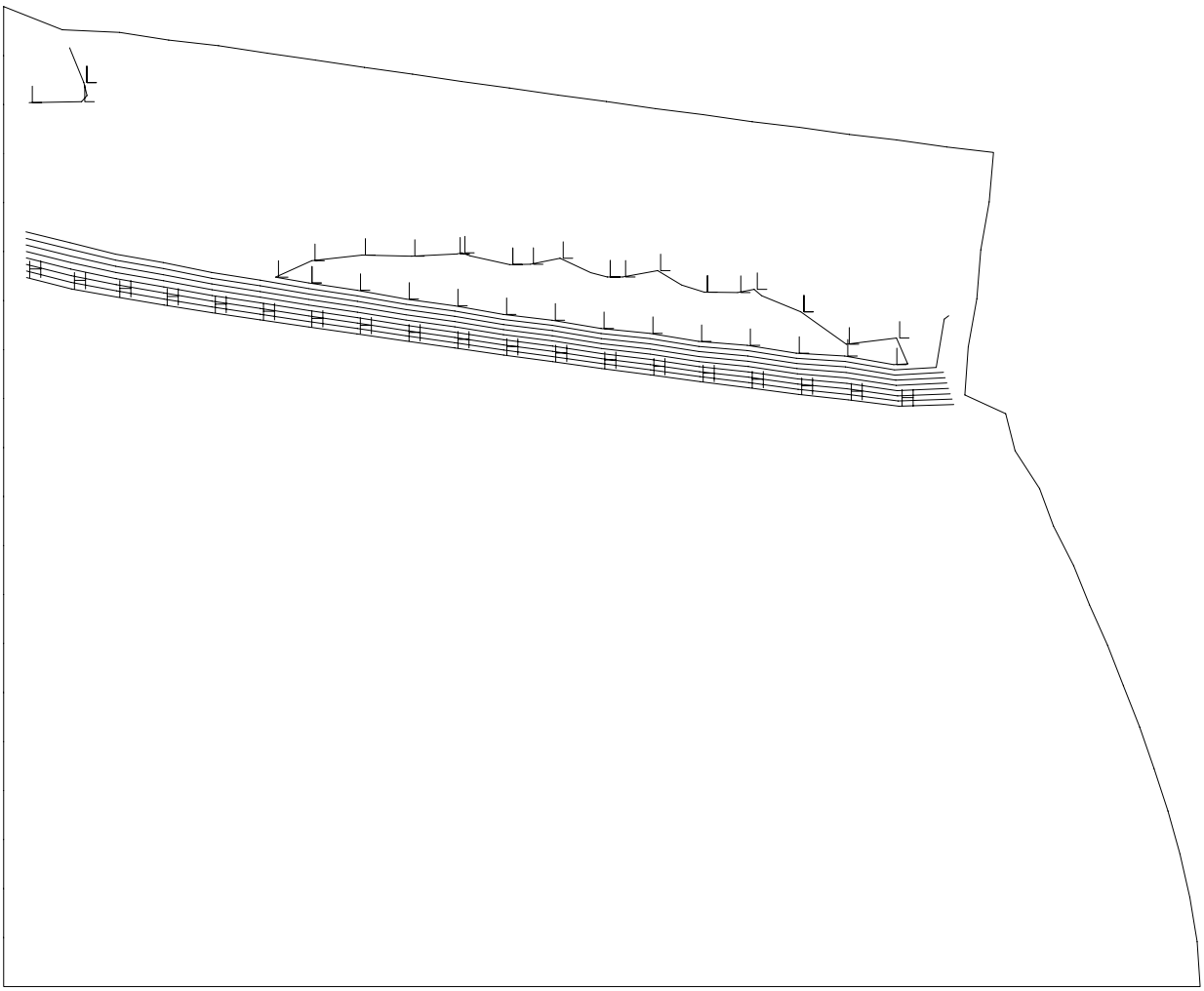


t= 2.00003E+01 cycle 6819

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

pressure min=−1.04018E+08 max= 8.64073E+07 L=−8.49752E+07 H= 6.73648E+07 dq= 1.90425E+07

contours for density

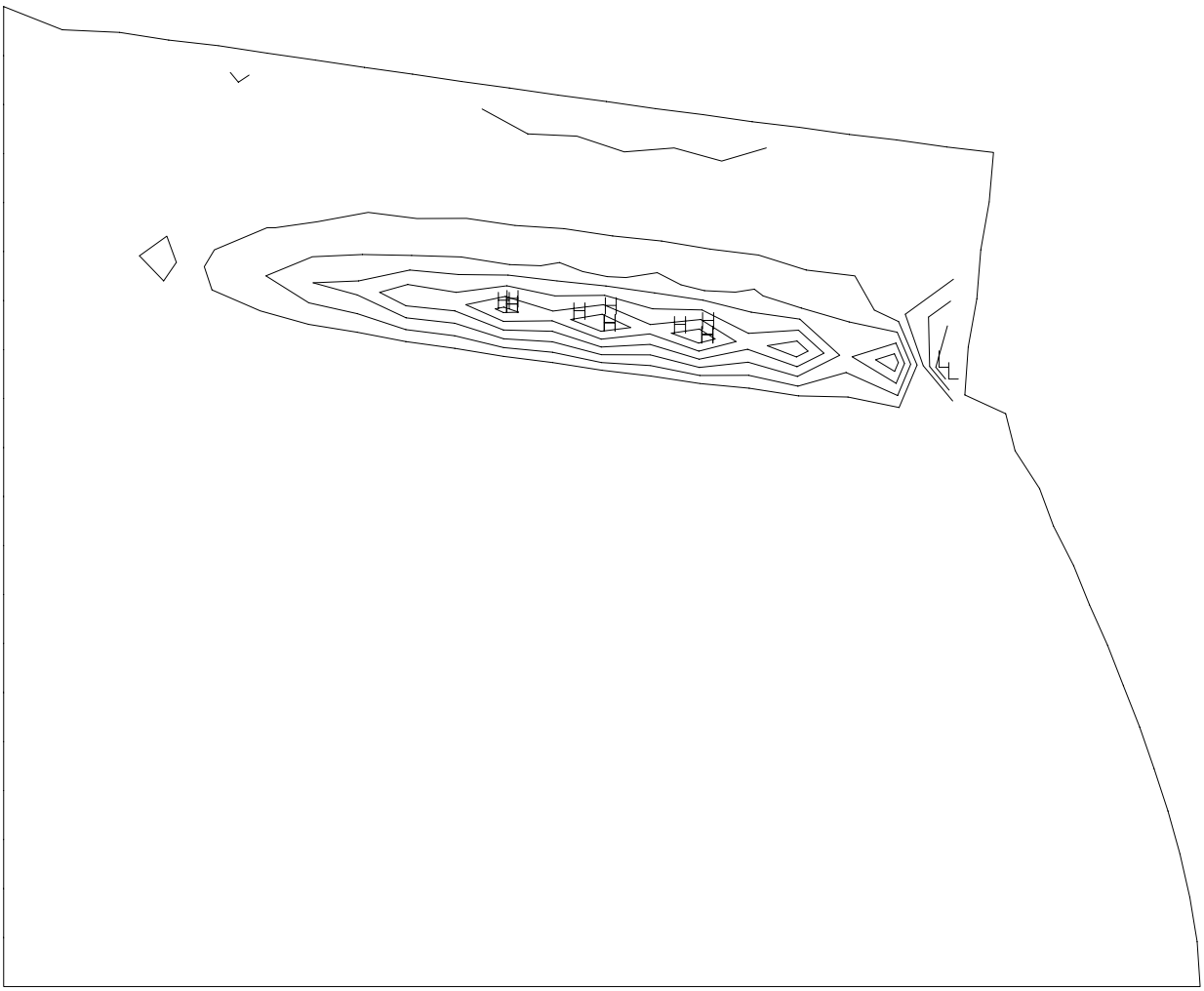


t= 2.00003E+01 cycle 6819

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

density min= 2.88338E+03 max= 3.00167E+03 L= 2.89521E+03 H= 2.98985E+03 dq= 1.18293E+01

contours for int.energy

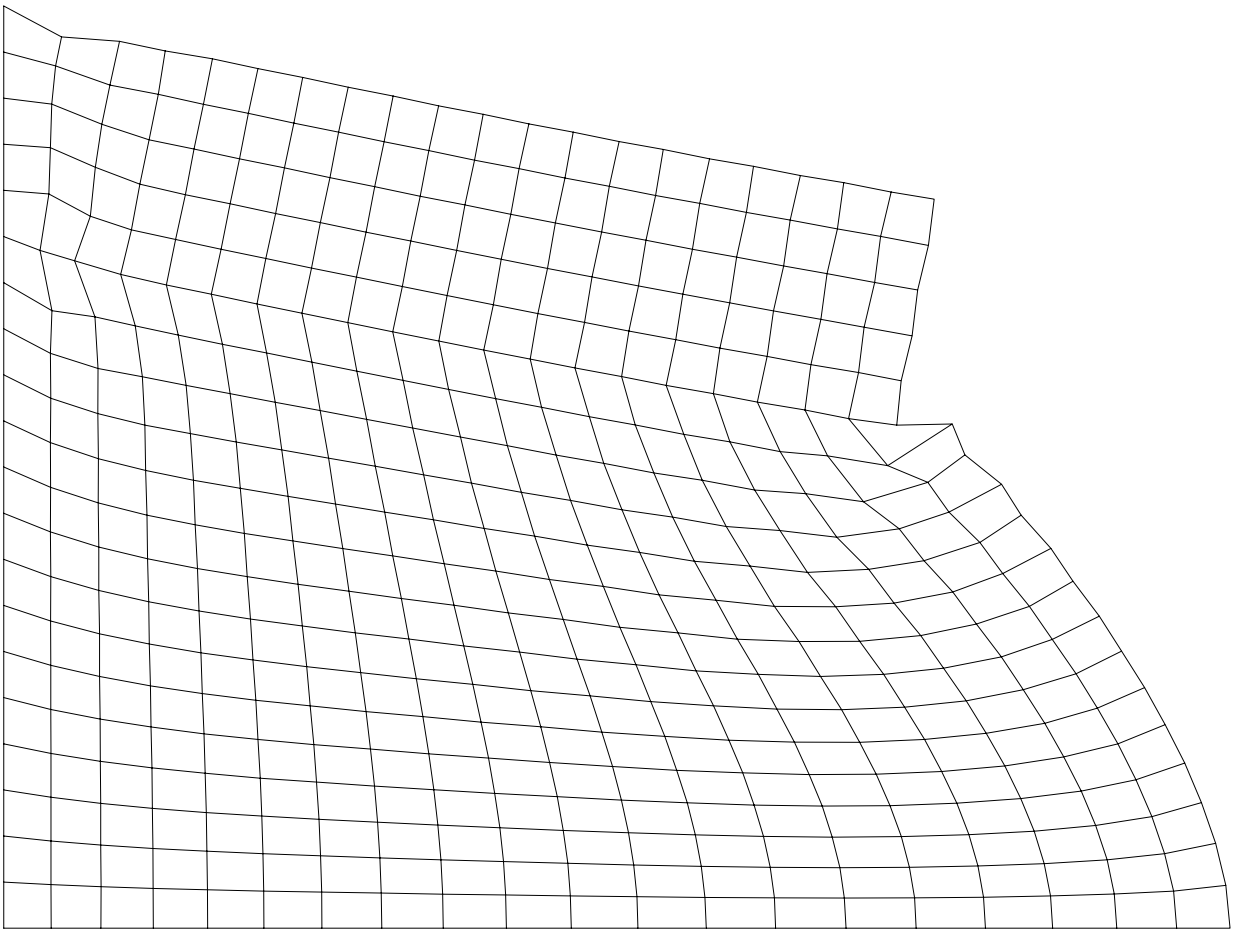


t= 2.00003E+01 cycle 6819

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

int.energy min=-1.09566E+05 max= 1.91378E+05 L=-7.94715E+04 H= 1.61283E+05 dq= 3.00944E+04

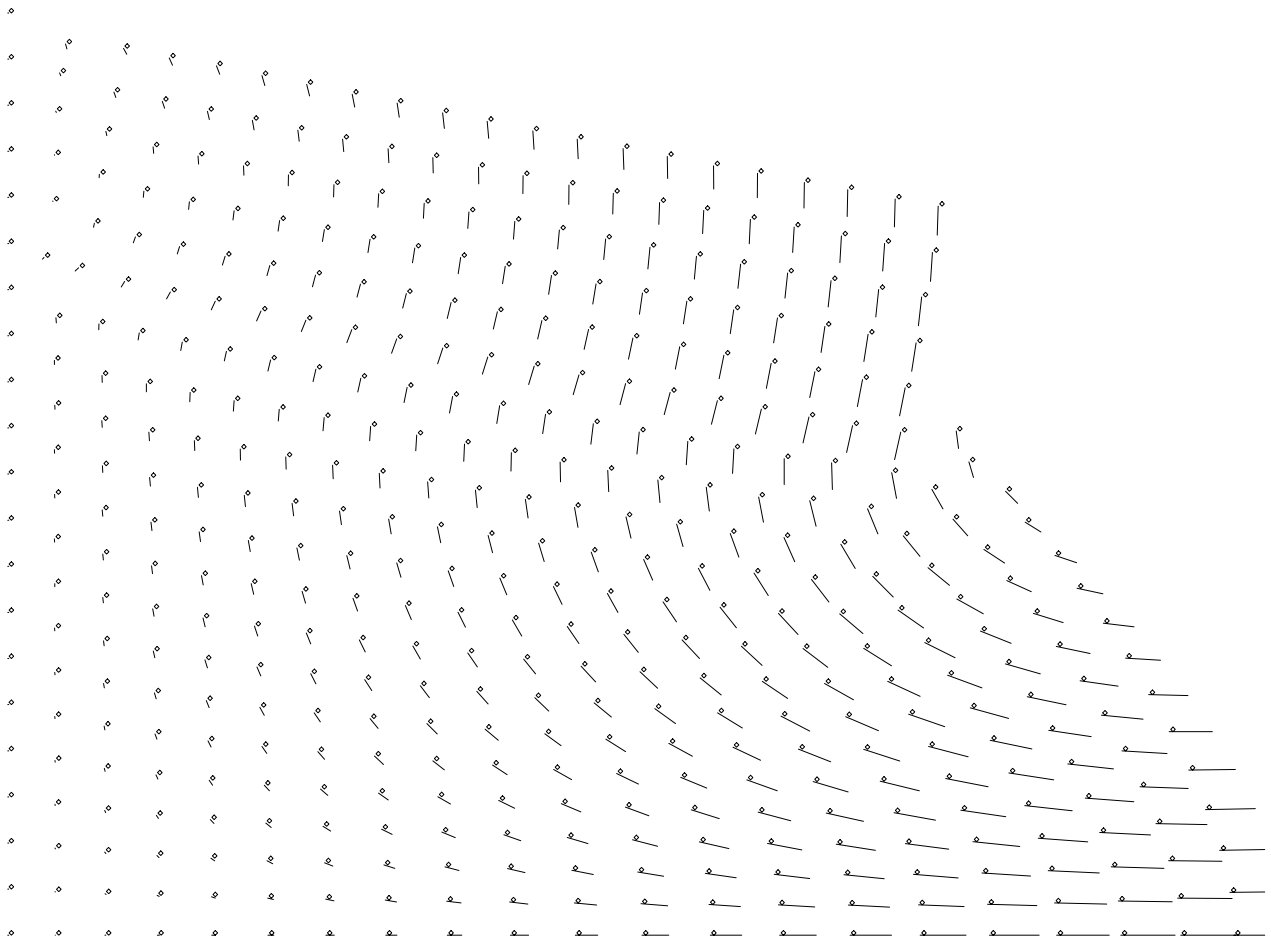
zone plot



t= 3.00012E+01 cycle 10738

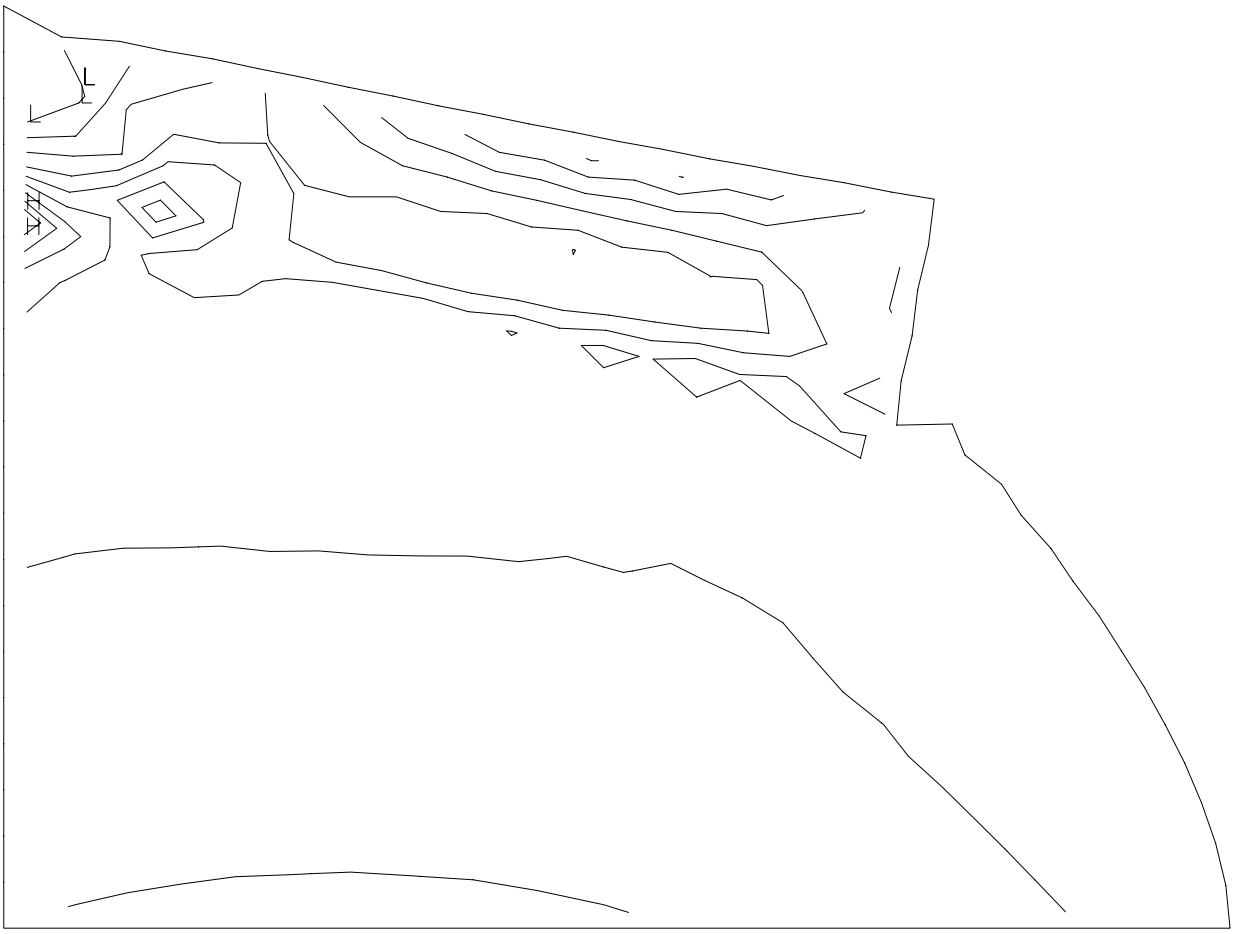
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

velplot



t= 3.00012E+01 cycle 10738 vmax= 2.07053E+01
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

contours for pressure

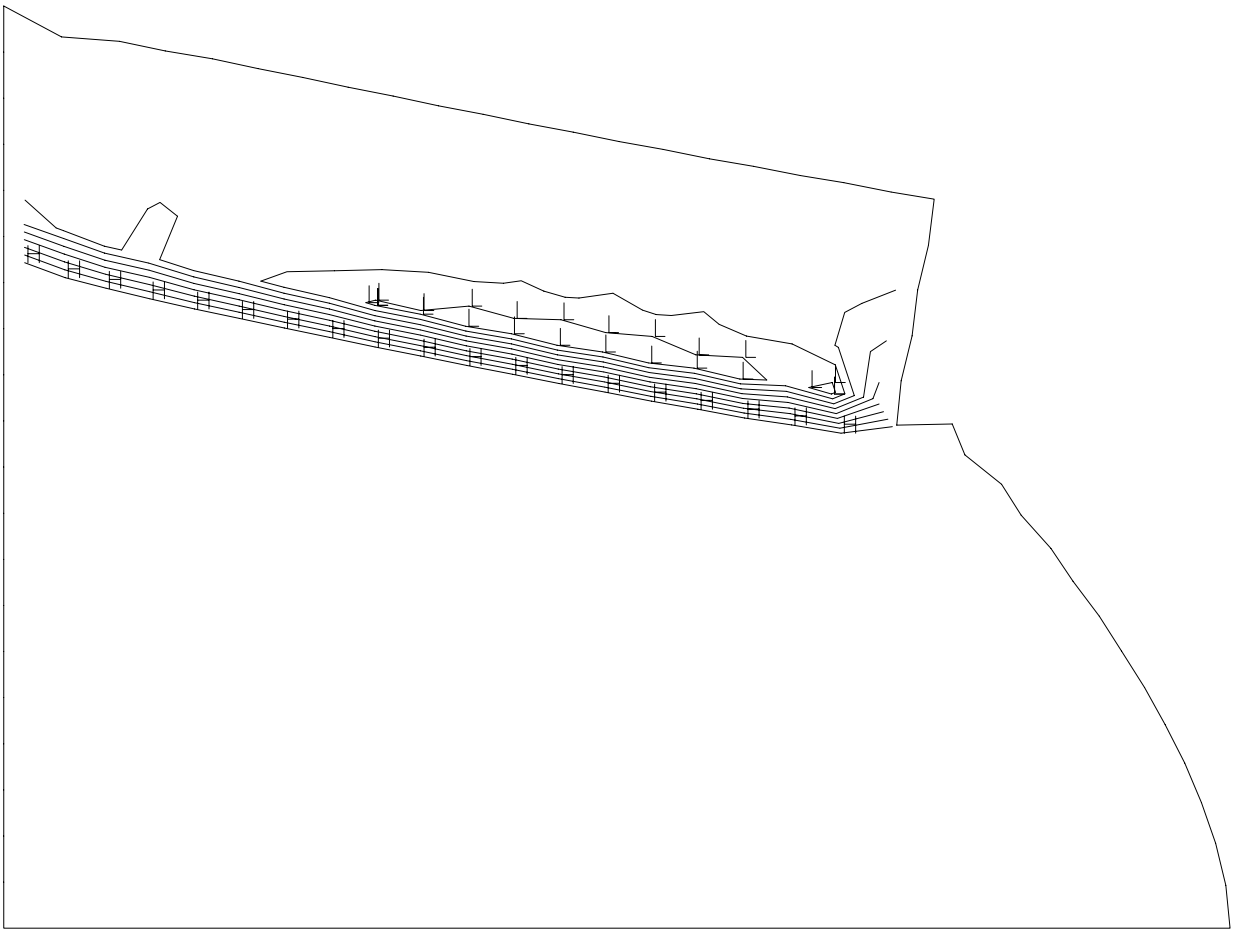


t= 3.00012E+01 cycle 10738

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

pressure min=−9.50320E+07 max= 8.21996E+07 L=−7.73089E+07 H= 6.44764E+07 dq= 1.77232E+07

contours for density

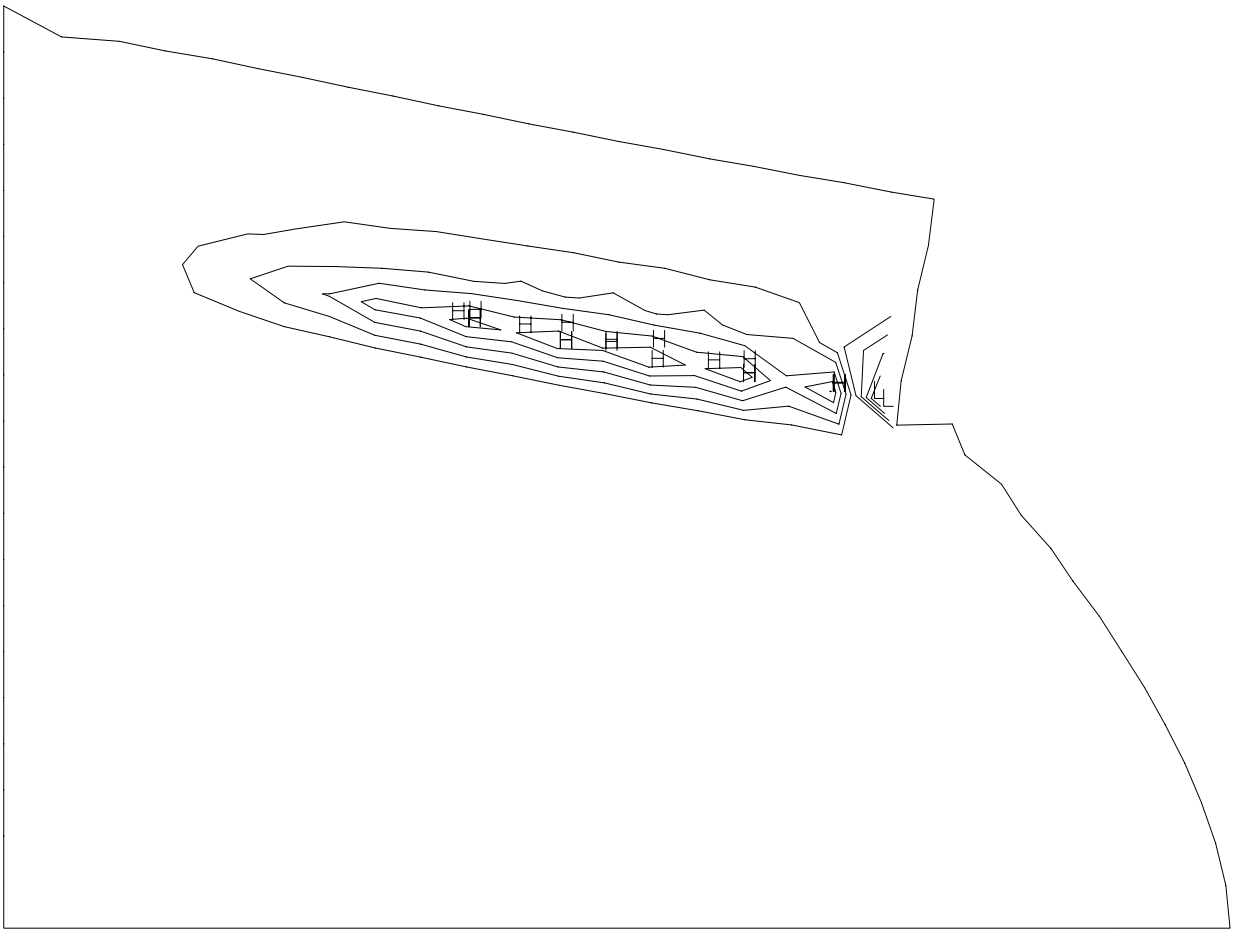


t= 3.00012E+01 cycle 10738

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

density min= 2.85950E+03 max= 3.00169E+03 L= 2.87371E+03 H= 2.98747E+03 dq= 1.42193E+01

contours for int.energy

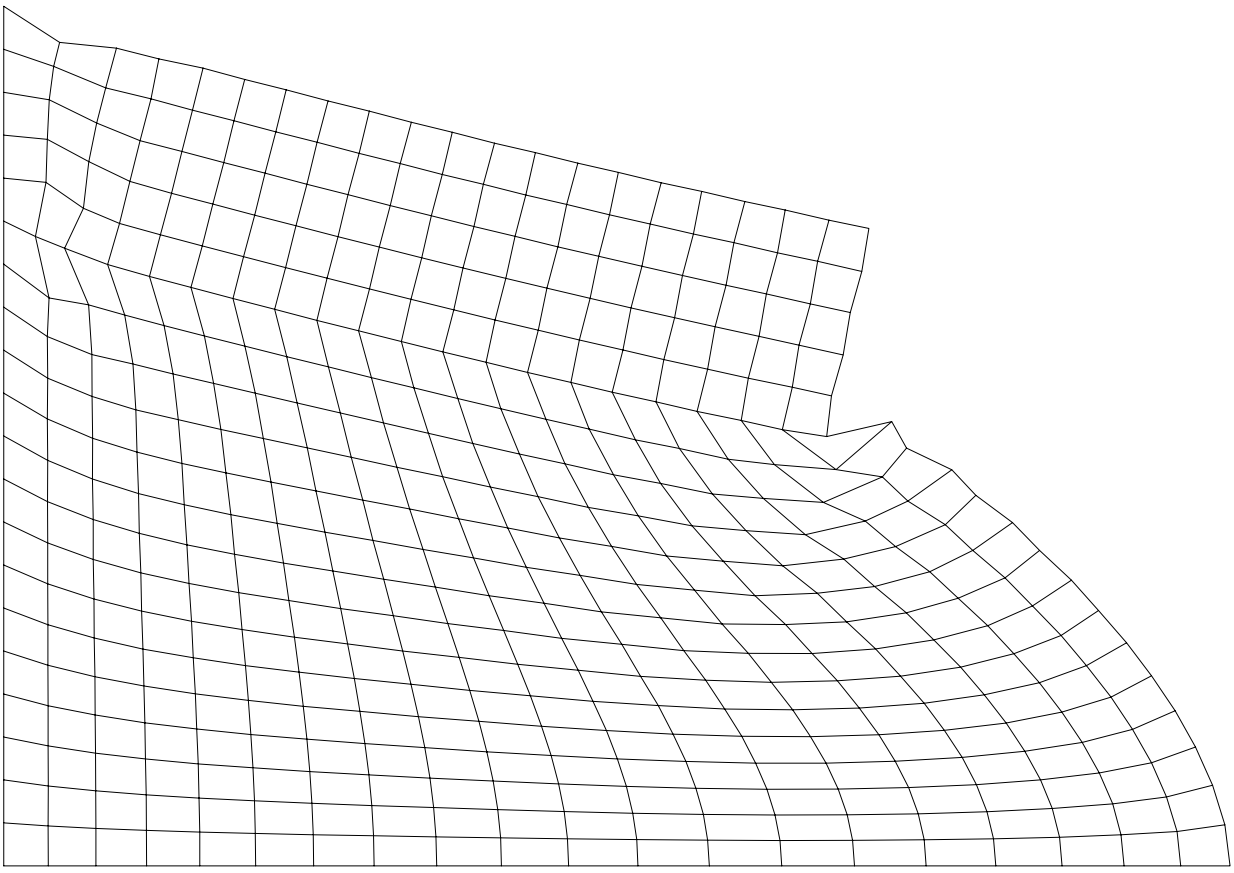


t= 3.00012E+01 cycle 10738

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

int.energy min=−4.62173E+05 max= 5.17730E+05 L=−3.64182E+05 H= 4.19740E+05 dq= 9.79903E+04

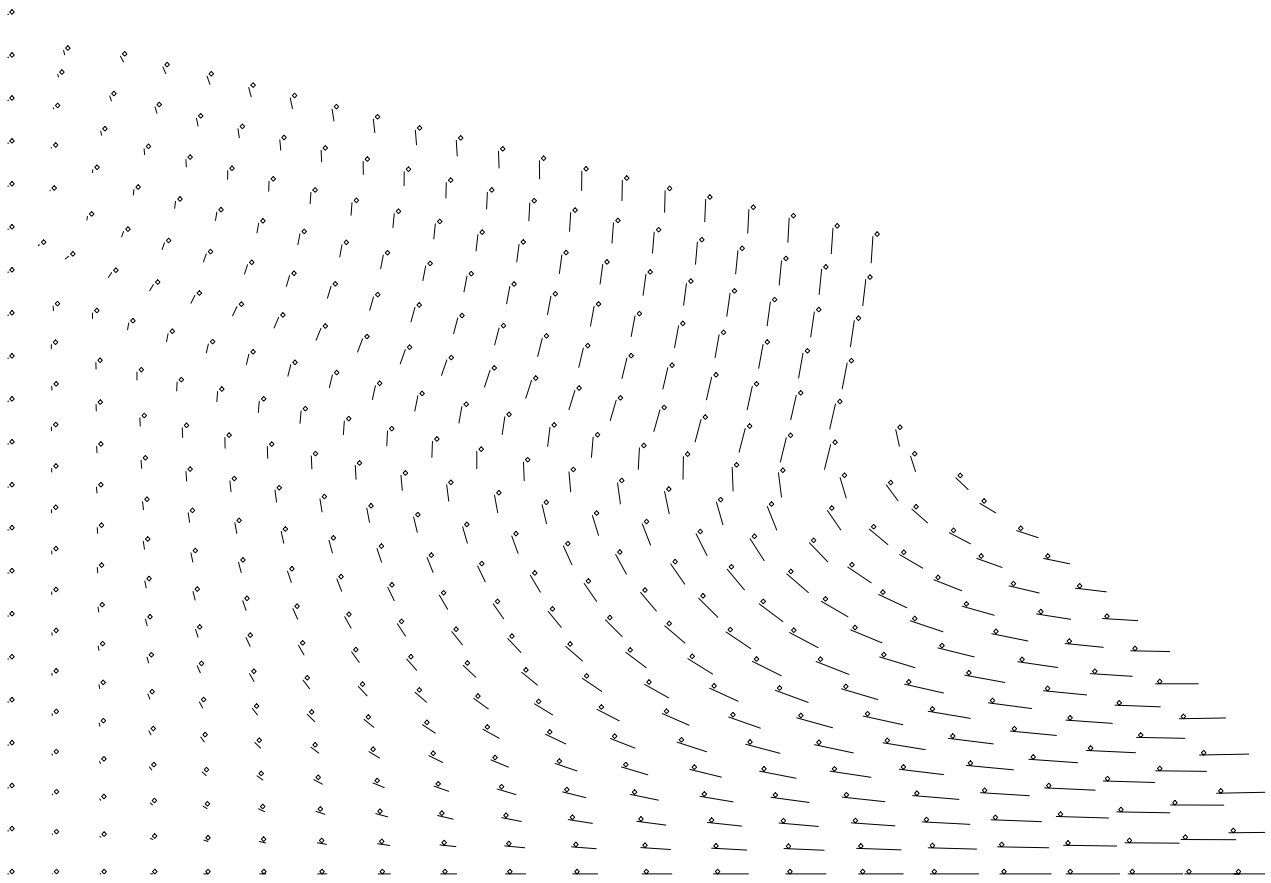
zone plot



t= 4.00020E+01 cycle 15370

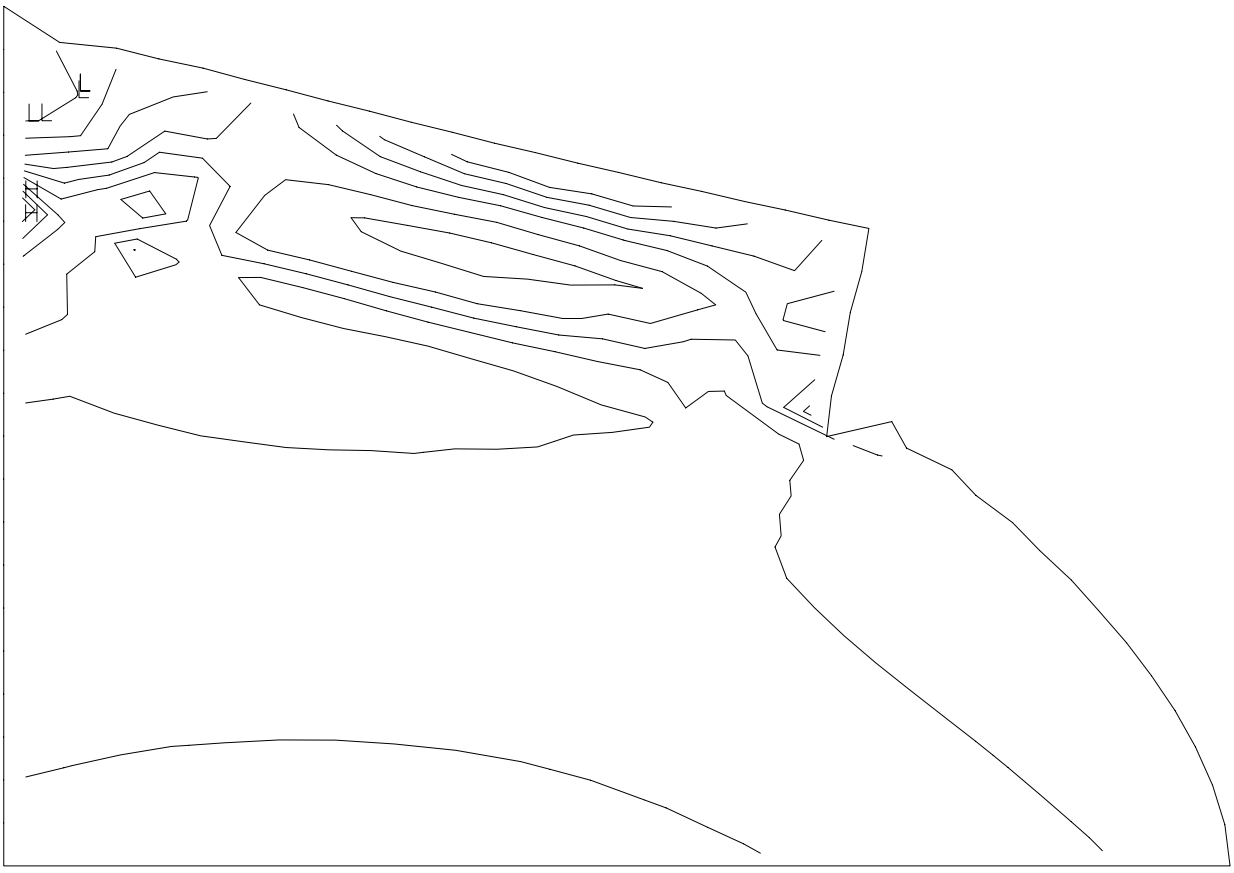
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

velplot



t= 4.00020E+01 cycle 15370 vmax= 1.82719E+01
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

contours for pressure

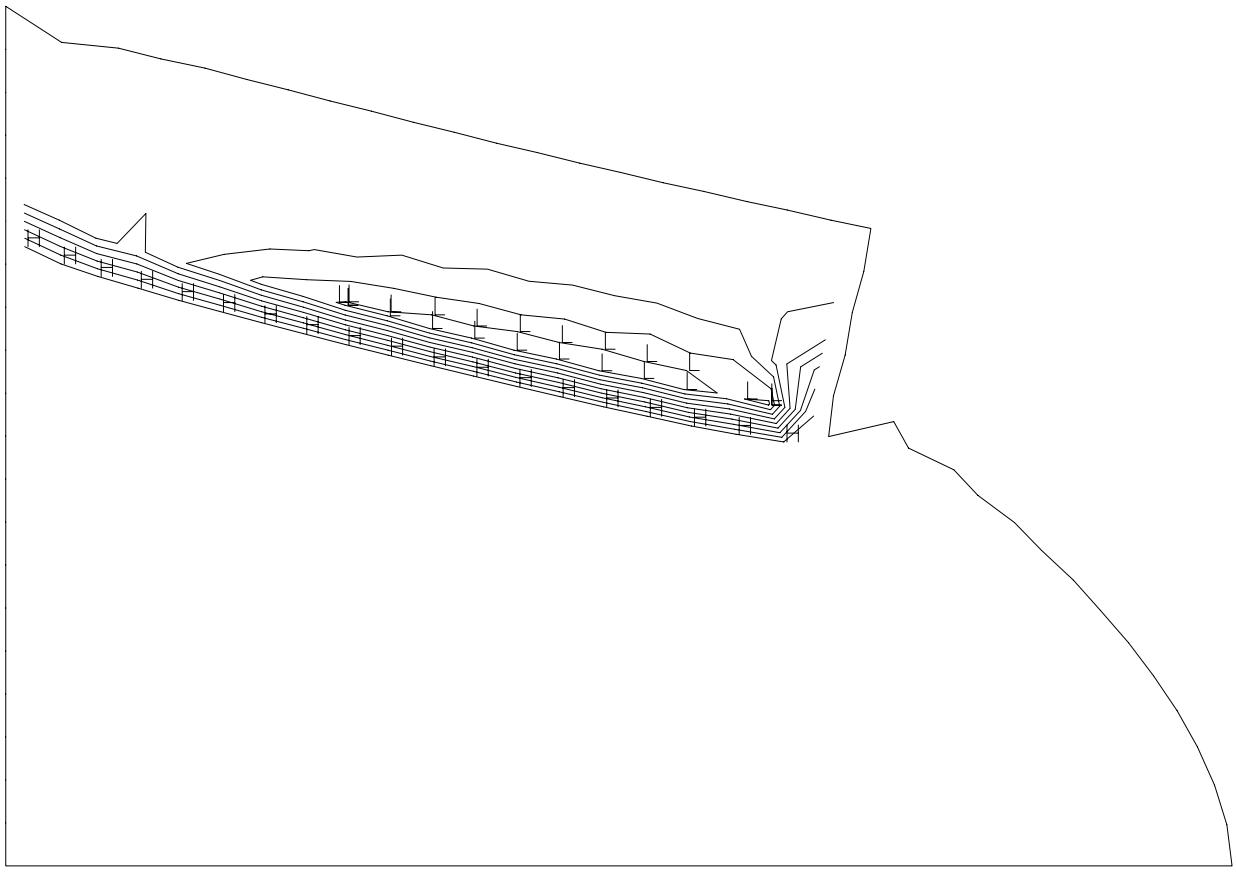


t= 4.00020E+01 cycle 15370

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

pressure min=−9.36437E+07 max= 7.52215E+07 L=−7.67572E+07 H= 5.83350E+07 dq= 1.68865E+07

contours for density

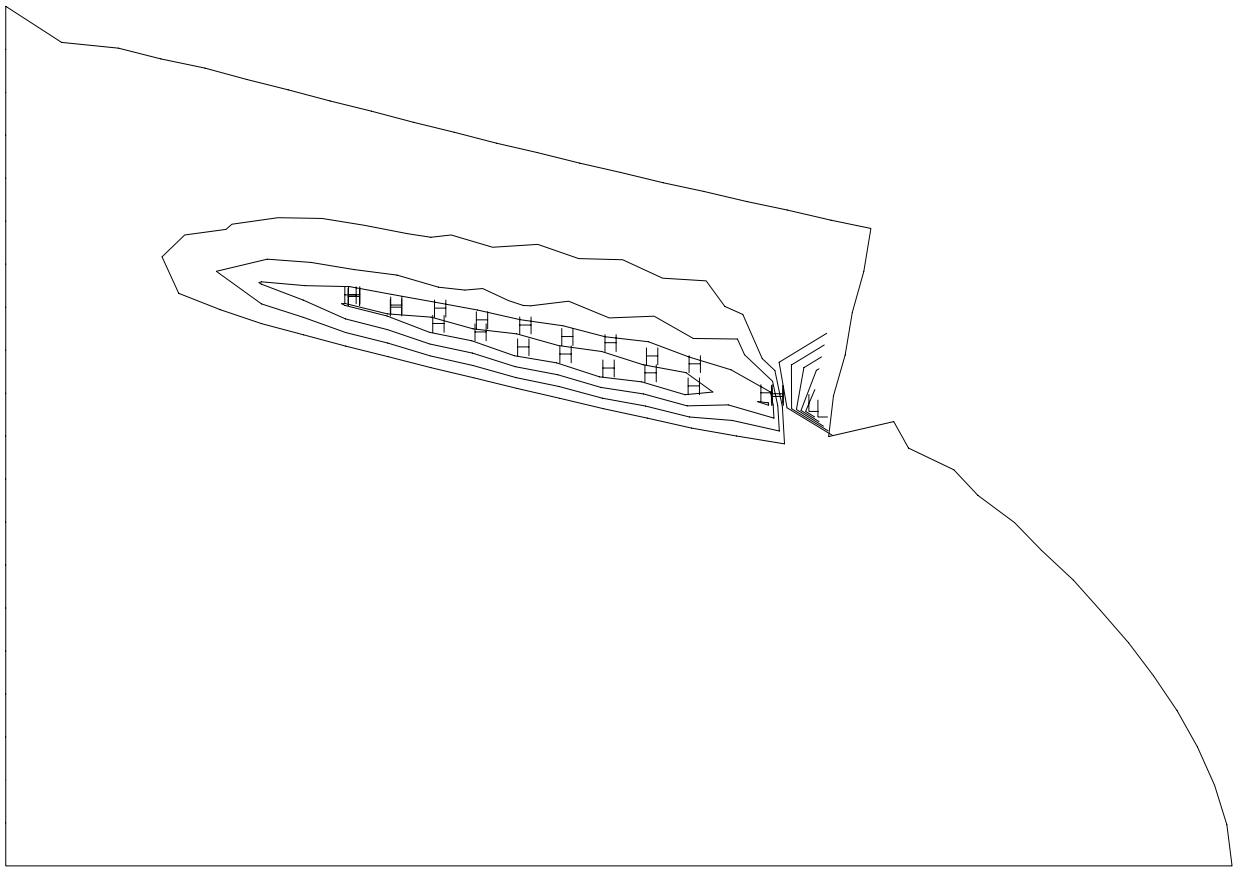


t= 4.00020E+01 cycle 15370

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

density min= 2.83825E+03 max= 3.00170E+03 L= 2.85460E+03 H= 2.98536E+03 dq= 1.63455E+01

contours for int.energy

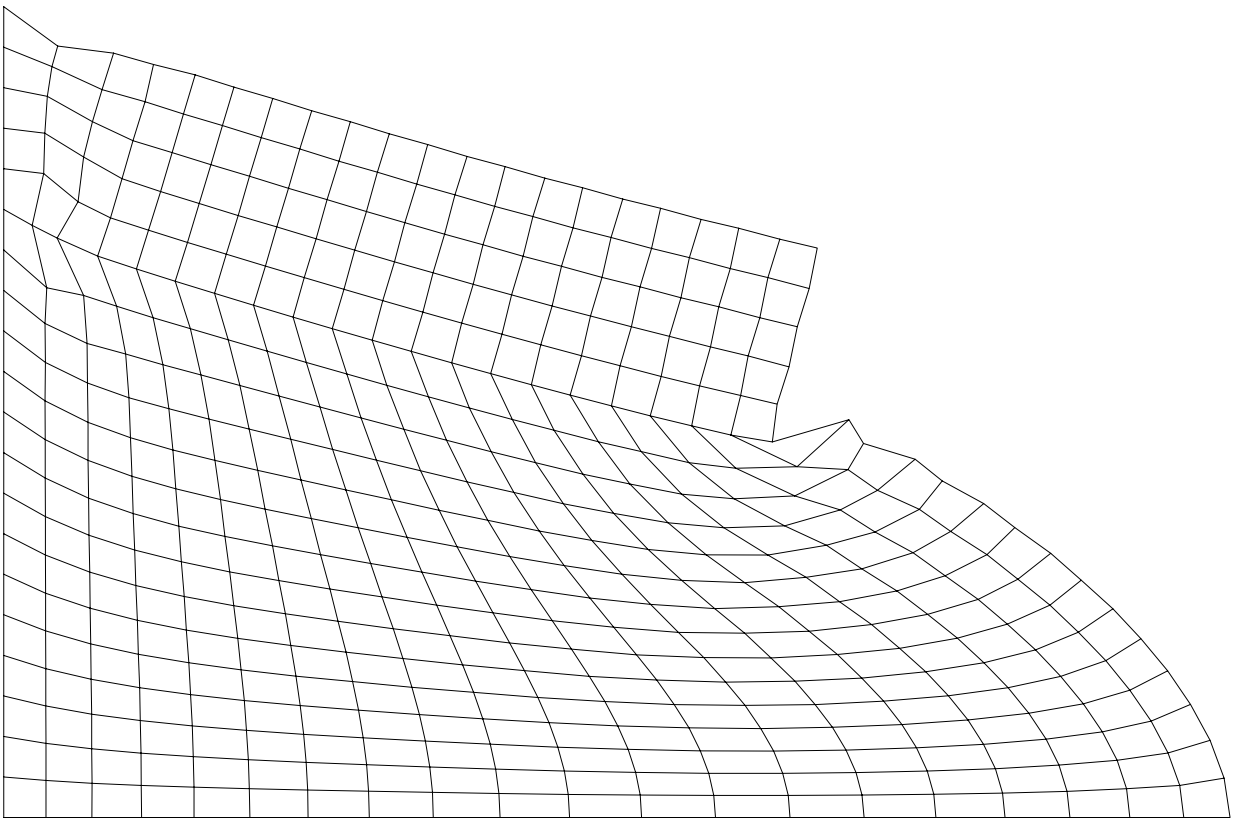


t= 4.00020E+01 cycle 15370

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

int.energy min=-1.08119E+06 max= 7.86435E+05 L=-8.94424E+05 H= 5.99673E+05 dq= 1.86762E+05

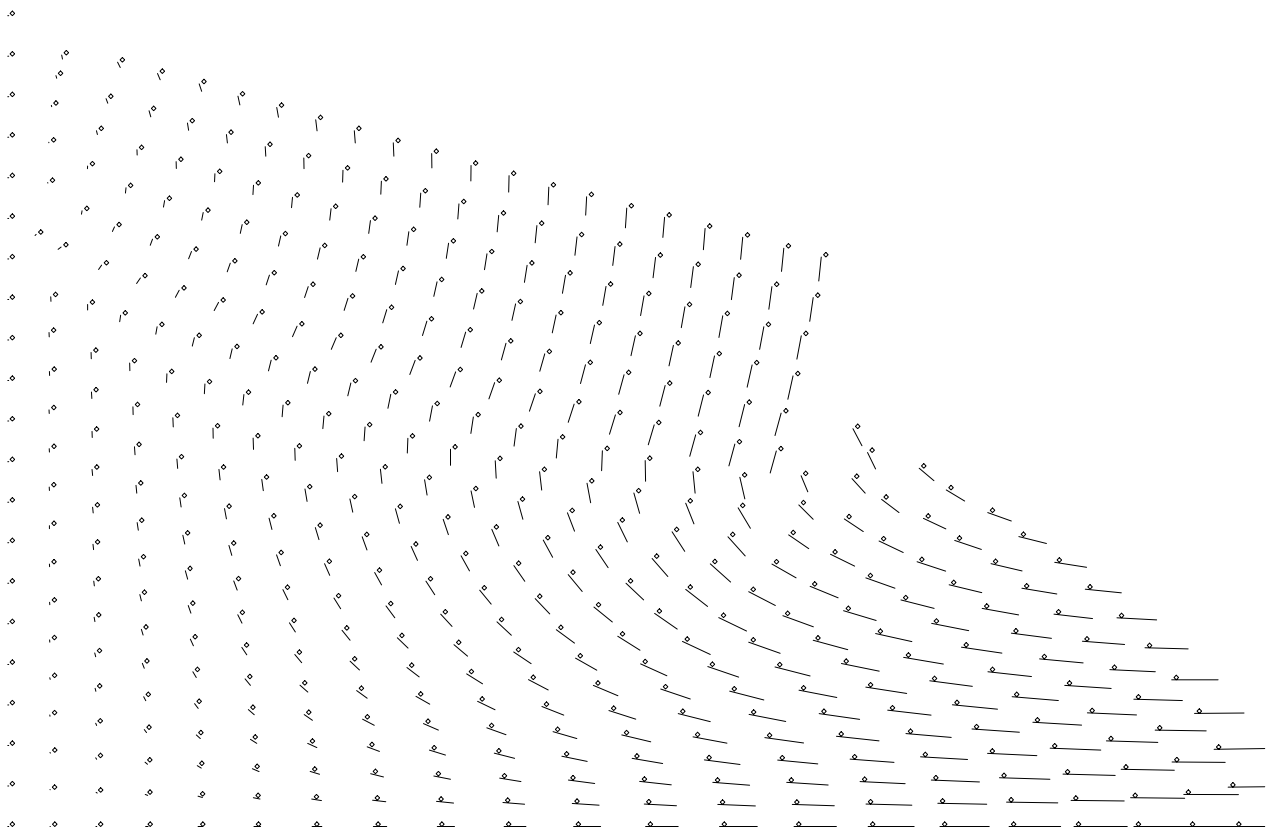
zone plot



t= 5.00008E+01 cycle 20739

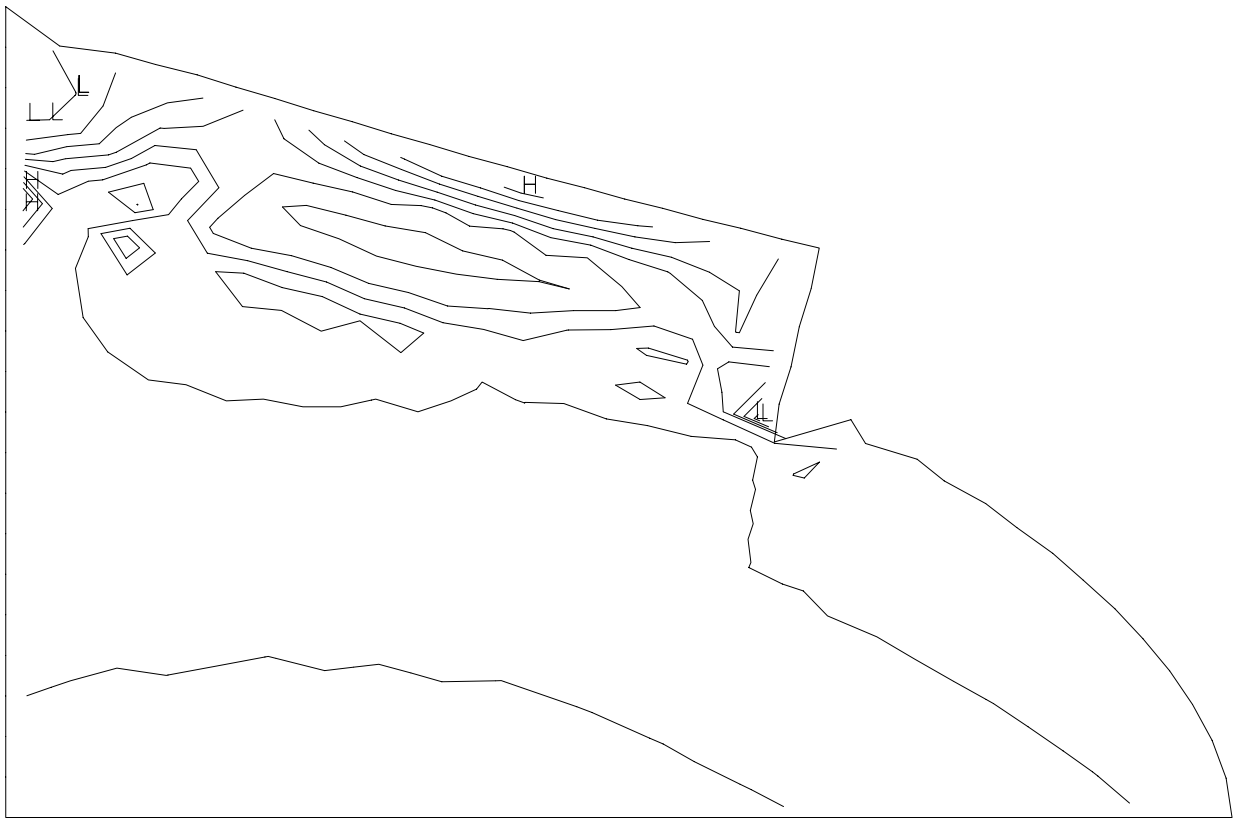
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

velplot



t= 5.00008E+01 cycle 20739 vmax= 1.62642E+01
SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

contours for pressure

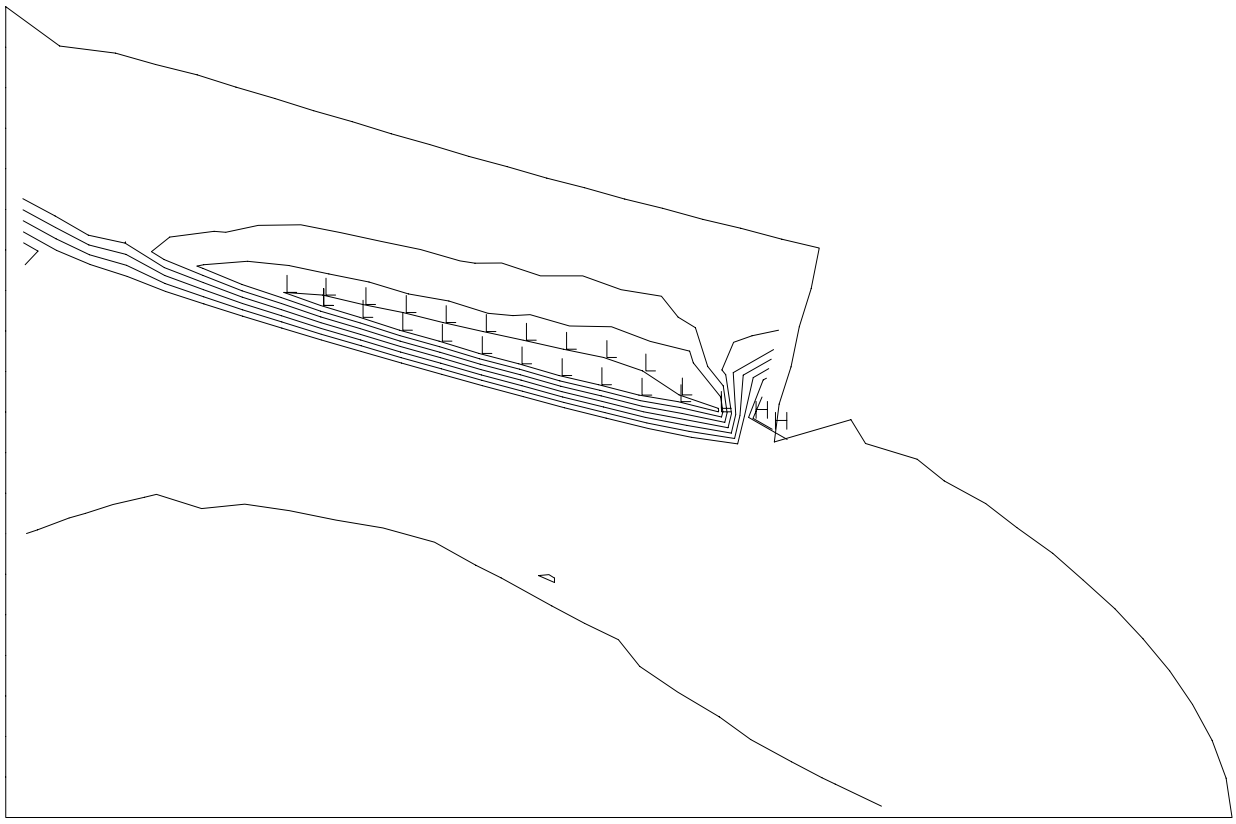


t= 5.00008E+01 cycle 20739

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

pressure min=−9.08060E+07 max= 7.05430E+07 L=−7.46711E+07 H= 5.44081E+07 dq= 1.61349E+07

contours for density

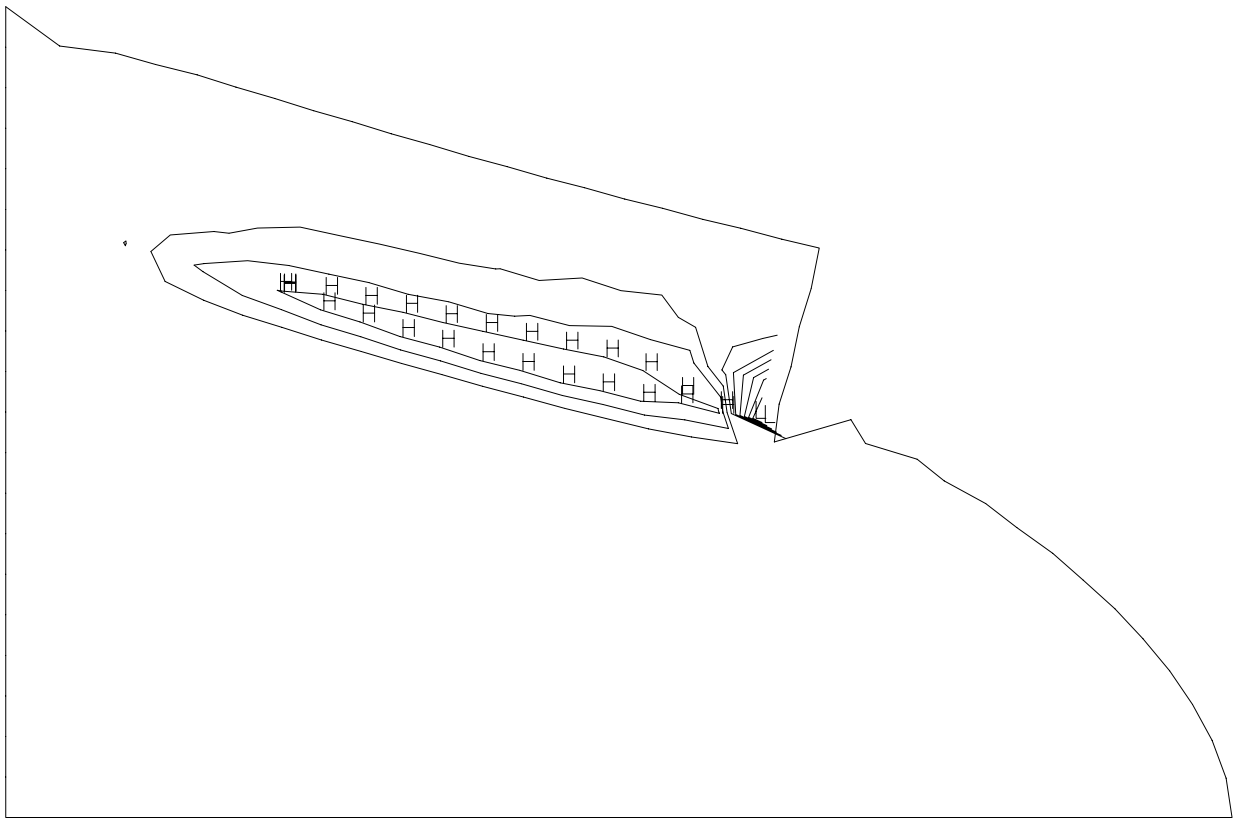


t= 5.00008E+01 cycle 20739

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

density min= 2.82402E+03 max= 3.04481E+03 L= 2.84610E+03 H= 3.02273E+03 dq= 2.20791E+01

contours for int.energy



t= 5.00008E+01 cycle 20739

SALES_2 v2.1 Fri Aug 2 13:07:50 2002 Example Problem: Collapsing Cliff Face

int.energy min=−1.87595E+06 max= 9.55879E+05 L=−1.59277E+06 H= 6.72696E+05 dq= 2.83183E+05